

Phase-Based Application-Driven Power Management on the Single-chip Cloud Computer

Nikolas Ioannou[†], Michael Kauschke[‡], Matthias Gries[‡], and
Marcelo Cintra^{†‡}

[†]University of Edinburgh

[‡]Intel Labs Braunschweig



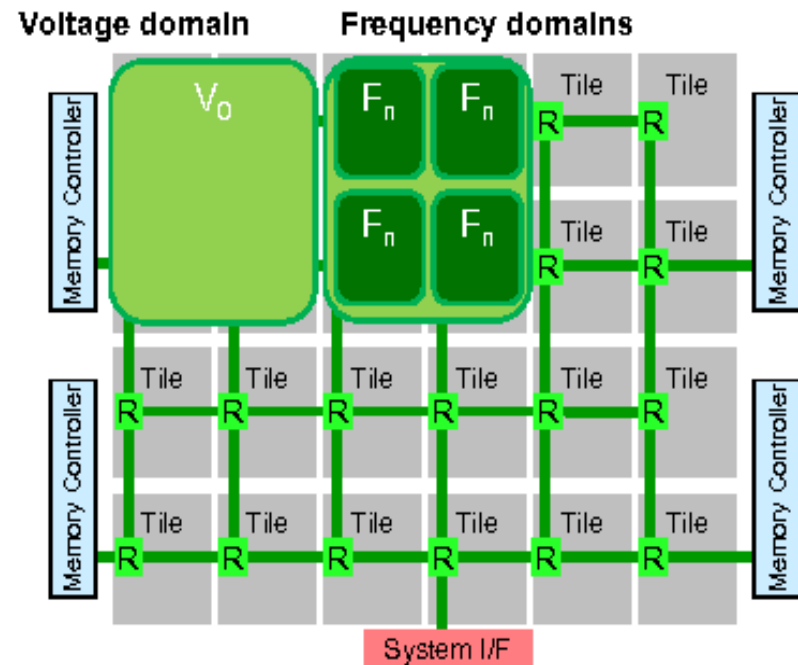
Introduction – Power Management

- Power and energy now first-order constraints
- Recent CPUs allow voltage/frequency scaling
- Minimize energy or power within some performance constraint
- Application Driven DVFS
 - Performance during memory-bound periods more or less unaffected by frequency
 - Reducing frequency and voltage saves energy/power
 - Periods are often recurrent
 - Periods can be learned and predicted



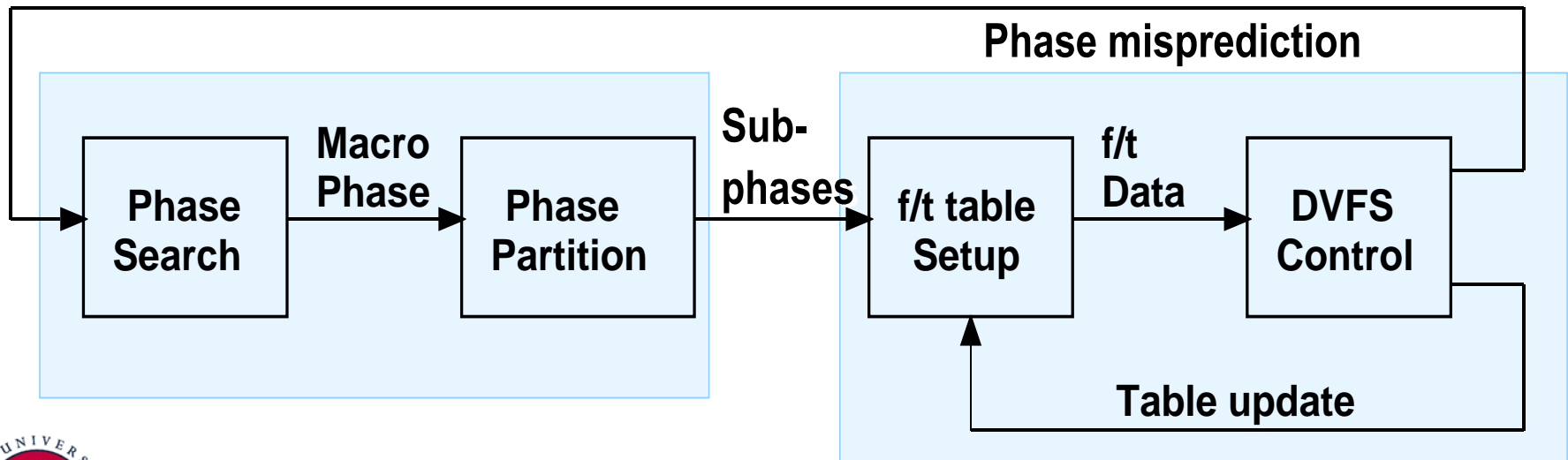
Introduction

- Dynamic Voltage and Frequency Scaling (DVFS) on Many-Cores
 - Voltage and frequency control not independent for each core: control \rightarrow granularity of domains
 - Possible application-level monitoring and control of settings
- The challenge
 - Power management on many-cores
 - Exploit application behavior to: Minimize energy consumption within a performance window
 - Target platform: SCC (exp. concept vehicle from Intel Labs) running MPI applications



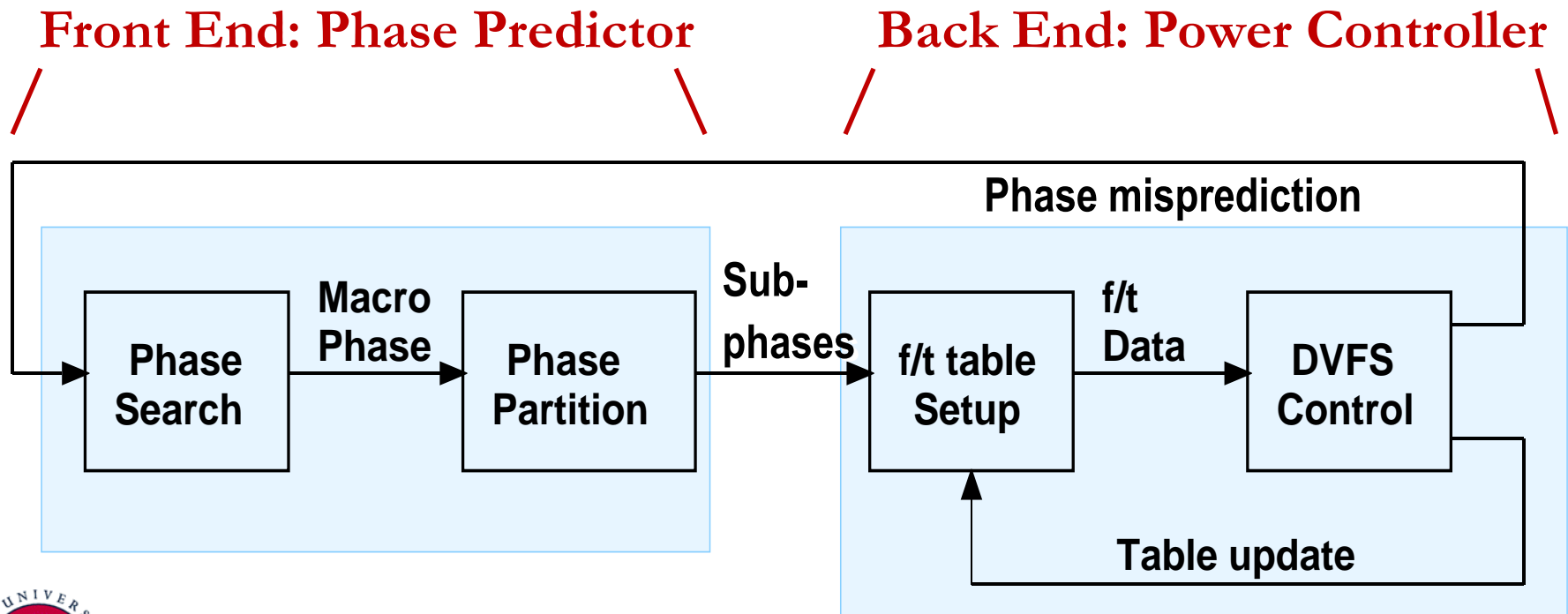
Proposed Scheme – Overview

- A modular, hierarchical, transparent, dynamic software power management scheme for a many-core system



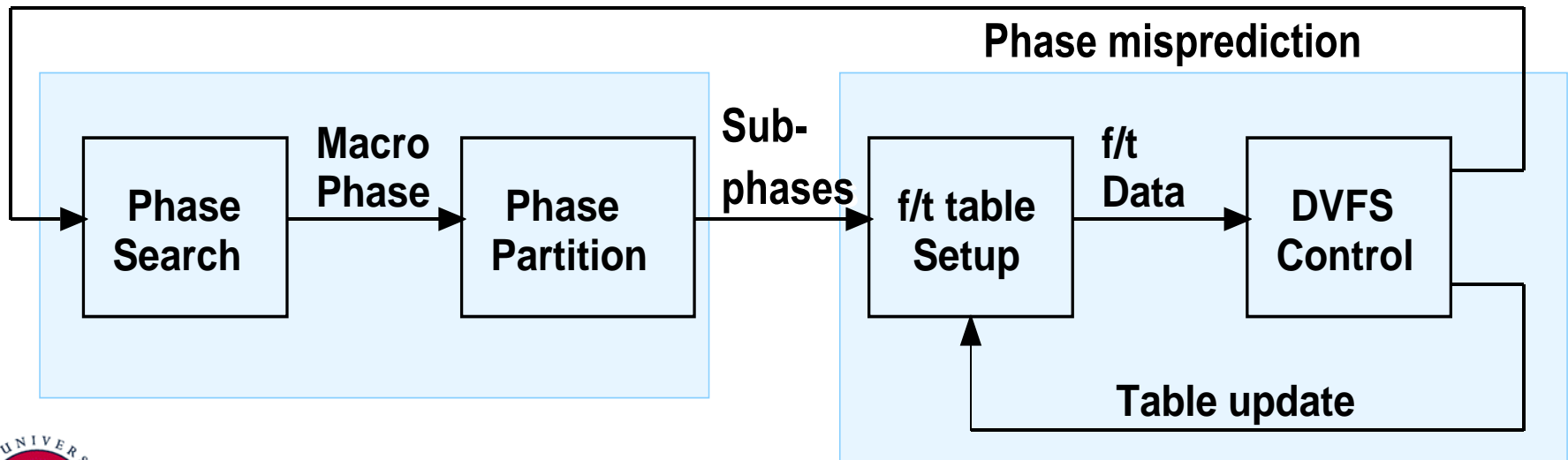
Proposed Scheme – Overview

- A modular hierarchical , transparent , dynamic software power management scheme for a many-core system



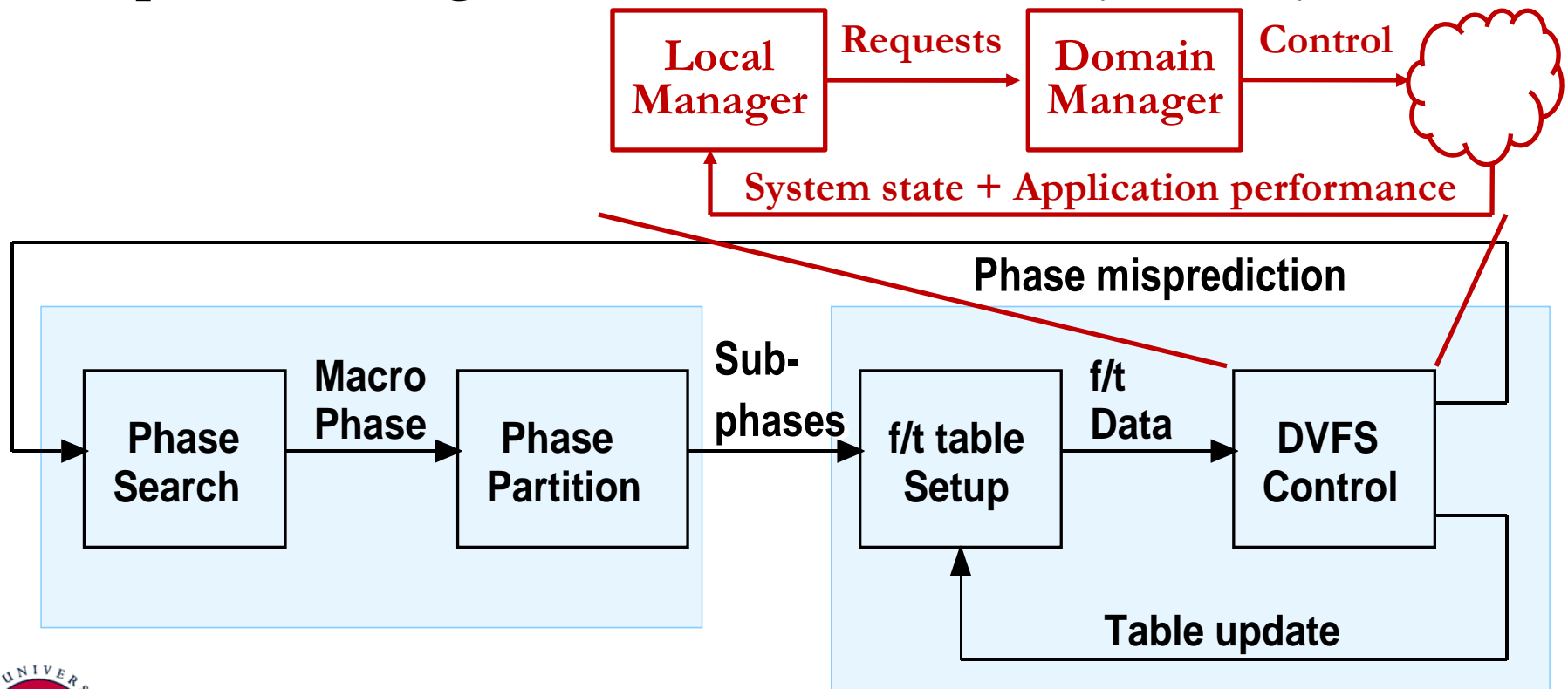
Proposed Scheme – Overview

- A modular, hierarchical, transparent, dynamic software power management scheme for a many-core system



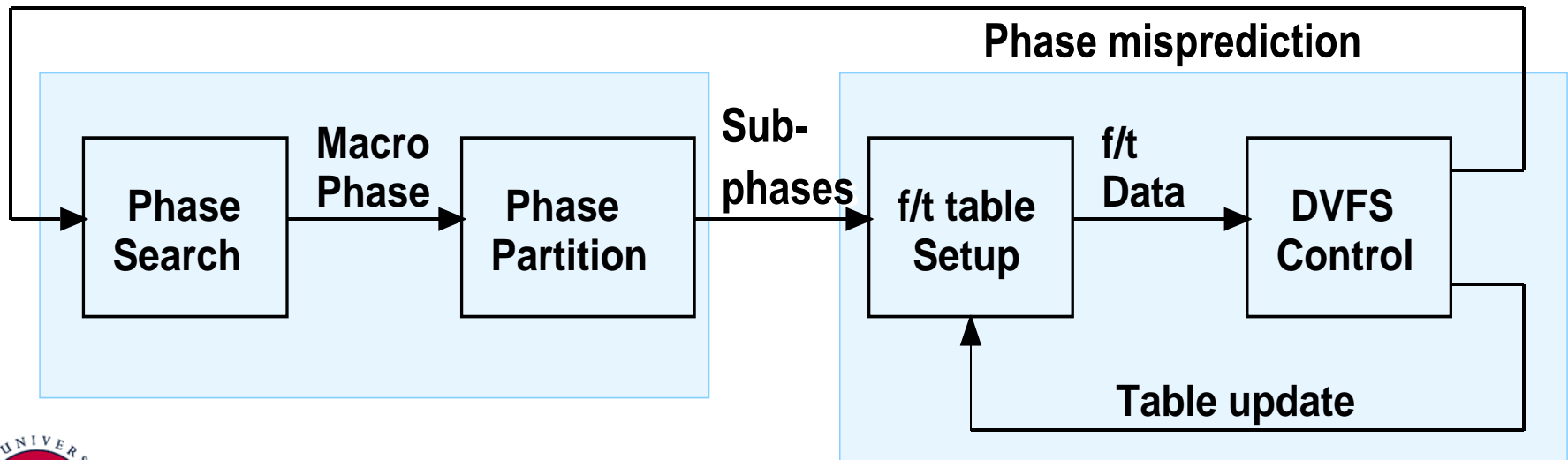
Proposed Scheme – Overview

- A modular, hierarchical, transparent, dynamic software power management scheme for a many-core system



Proposed Scheme – Overview

- A modular, hierarchical, transparent, dynamic software power management scheme for a many-core system

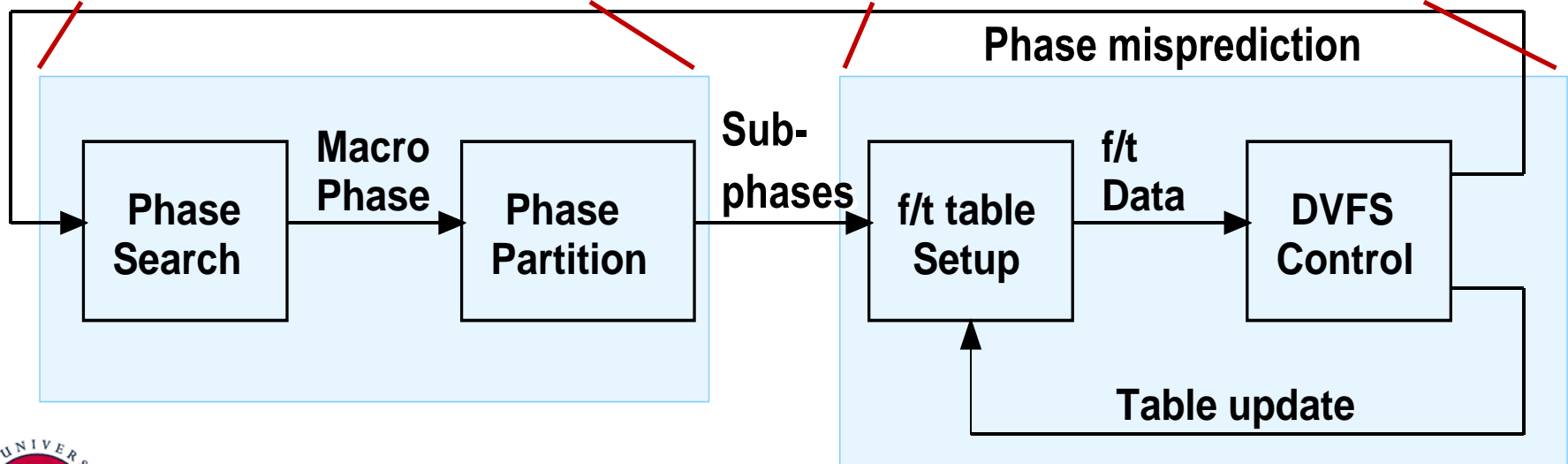


Proposed Scheme – Overview

- A modular, hierarchical, transparent, dynamic software power management scheme for a many-core system

Captures application behavior without user intervention

Tracks performance vs. energy behavior and adapts



Contributions

- Novel power management scheme for many-core systems
 - Hierarchical → capable of operating on domain-based systems
- Novel phase prediction scheme
(SMRP – SuperMaximal Repeat phase Predictor)
 - Based on supermaximal repeat string → better accuracy than previous approaches
 - Transparent instrumentation of MPI applications
- Implemented and evaluated schemes on a real experimental many-core system
- Significant energy savings with little performance degradation
 - Average 15% energy savings with only 7% performance degradation
 - Well within 3:1 ratio of power savings to performance degradation



Outline

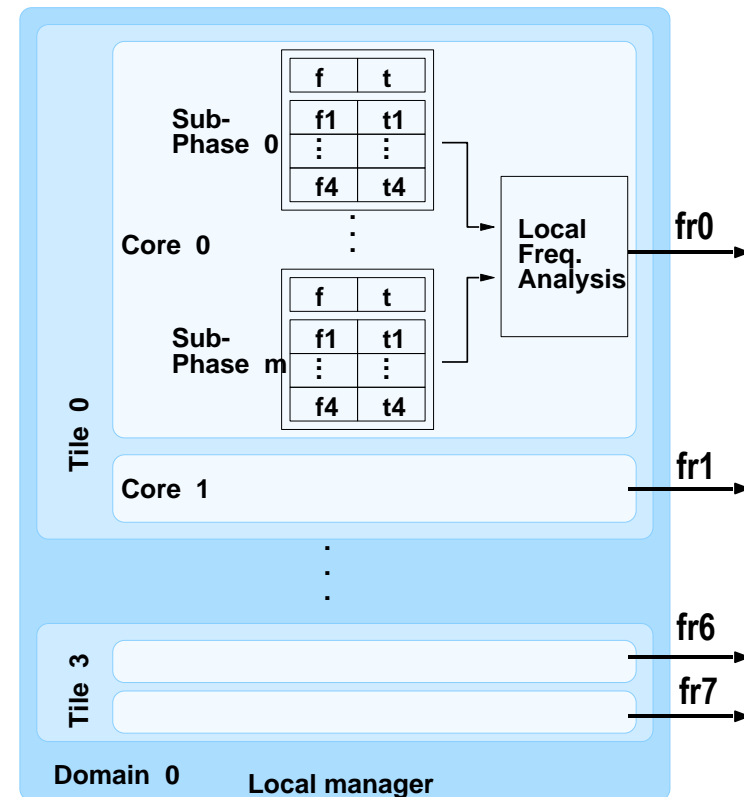
- Introduction
- Power Manager
- Phase Predictor
- Results
- Conclusions



Power Manager: Local Manager

- Input: phases of repeatable behavior
- For each repetitive phase a frequency/time table is built
- Iterative approach:

1. Start at highest frequency
2. Measure execution time of current instance and record in table
3. Reduce frequency by one step
4. Stop if performance impact higher than threshold $-\delta$ – (e.g., 10%)
5. Otherwise, repeat until lowest frequency is reached



▪ Output: frequency requested per core



Local Management Example

Learning →



Local Management Example

Learning →



Sub-phase i detected:

Set f to highest value (e.g., $f_i^l = 800\text{MHz}$)

Measure execution time t_i^l

Local Management Example

Learning →



Sub-phase i detected:

Set f to highest value (e.g., $f_i^1=800\text{MHz}$)

Measure execution time t_i^1

Sub-phase i detected:

Set f to next value (e.g., $f_i^2=533\text{MHz}$)

Measure execution time t_i^2

$t_i^2 < (1+\delta)t_i^1 \rightarrow$ continue exploration

Local Management Example

Learning →



Sub-phase i detected:

Set f to highest value (e.g., $f_i^1=800\text{MHz}$)

Measure execution time t_i^1

Sub-phase i detected:

Set f to next value (e.g., $f_i^2=533\text{MHz}$)

Measure execution time t_i^2

$t_i^2 < (1+\delta)t_i^1 \rightarrow$ continue exploration

Sub-phase i detected:

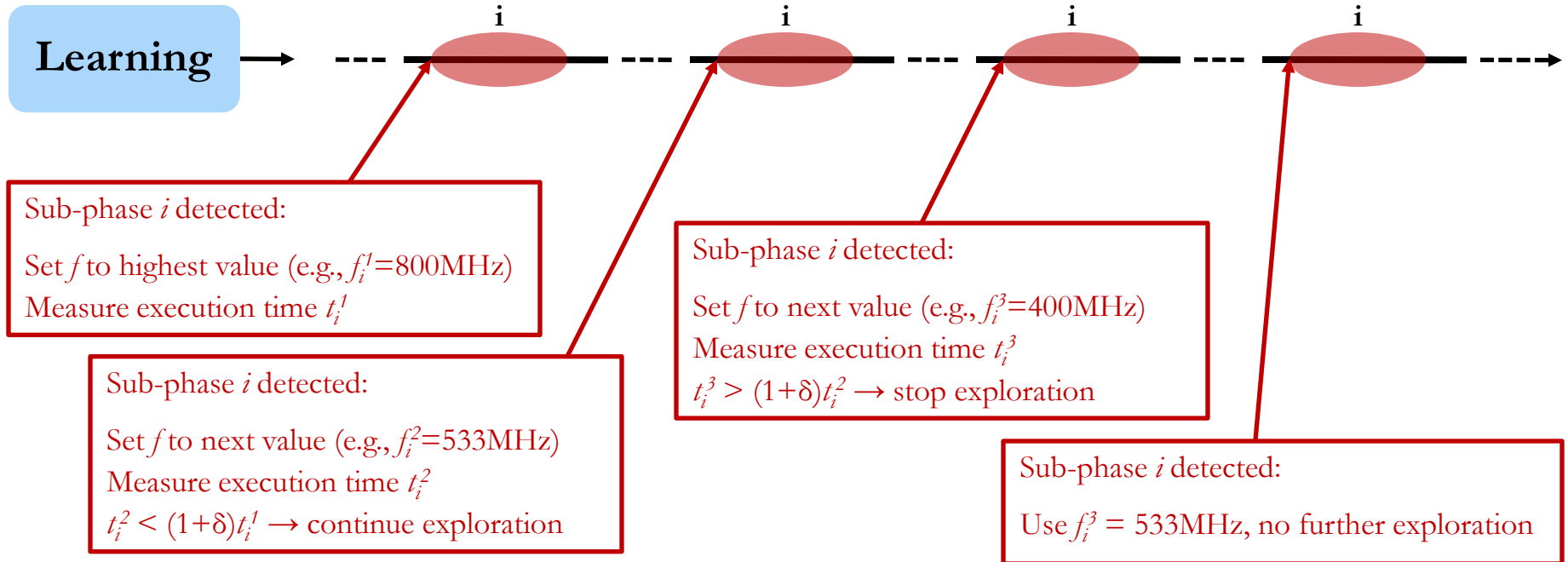
Set f to next value (e.g., $f_i^3=400\text{MHz}$)

Measure execution time t_i^3

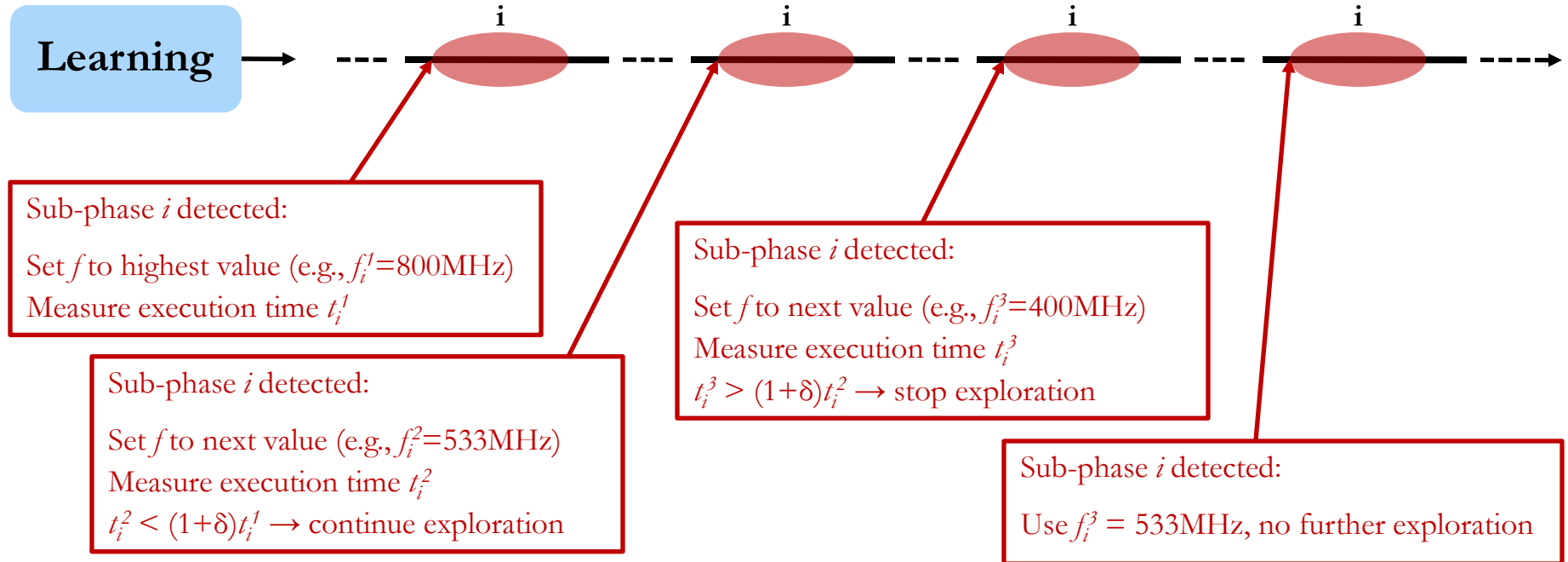
$t_i^3 > (1+\delta)t_i^2 \rightarrow$ stop exploration



Local Management Example



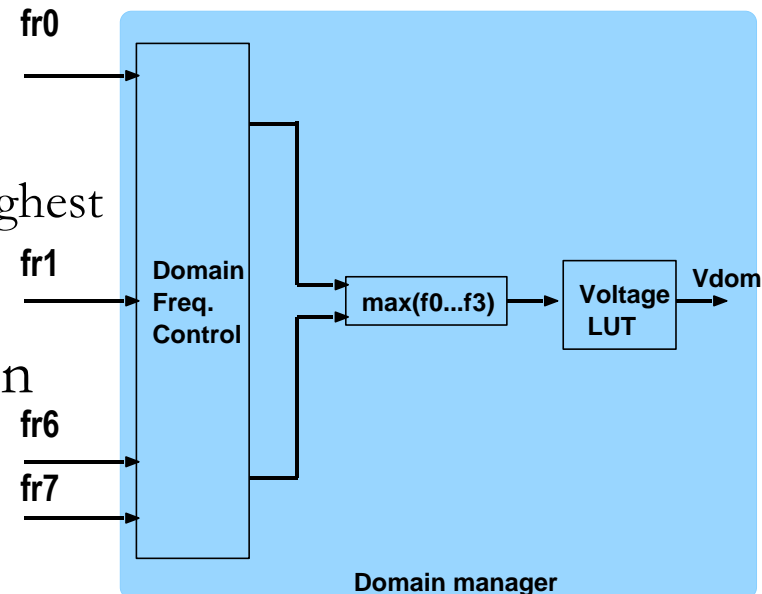
Local Management Example



- Steady-state frequency will change if there is a phase misprediction
- In reality test of new frequencies is dependent on domain management decisions

Power Manager: Domain Manager

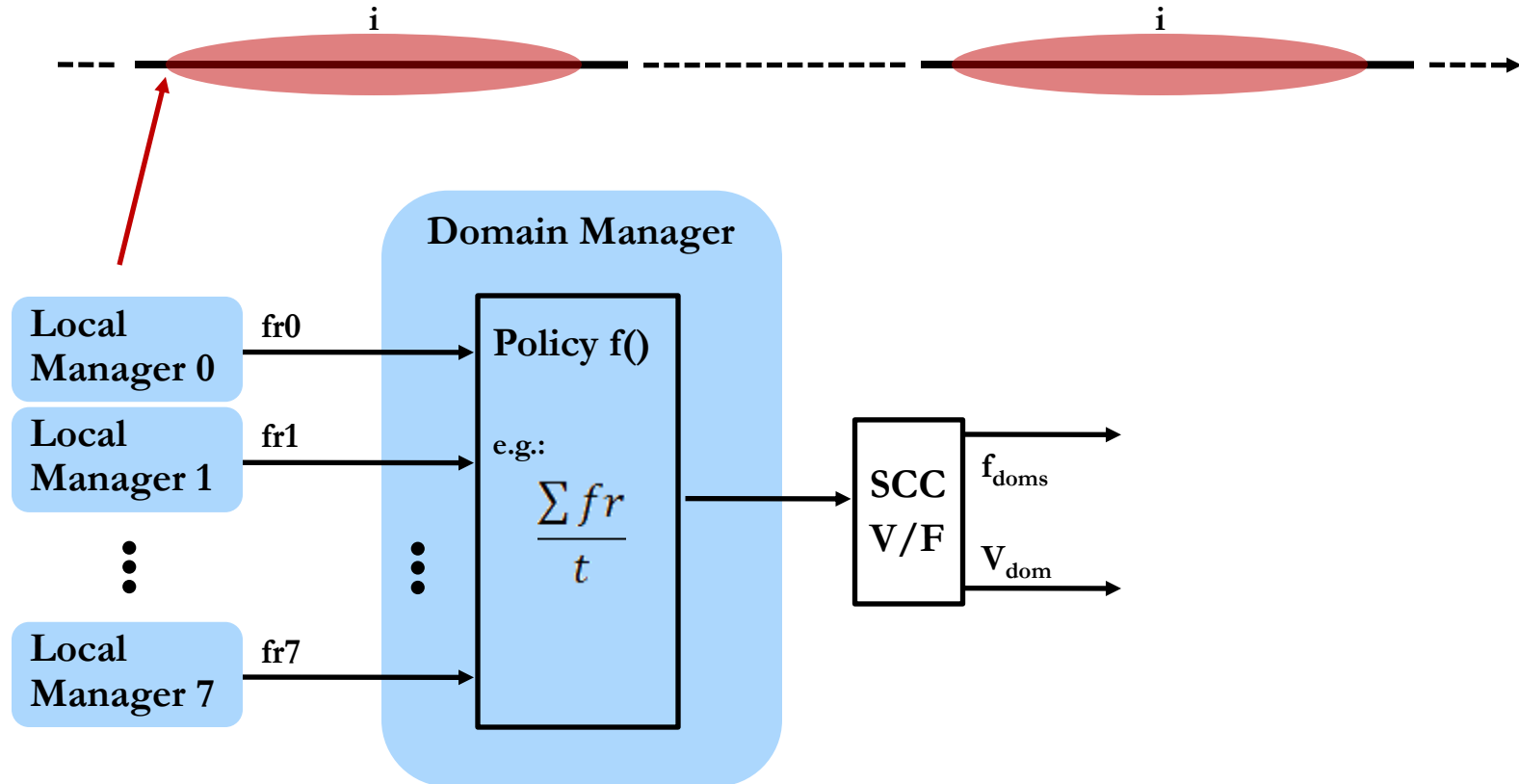
- One per voltage domain
- Decides/controls voltage and frequency
- Input: frequency requests from local managers for each core
- Output: frequencies for the entire domain. Policies investigated:
 - **Simple:** simply service requests in order
 - **Mean:** select the mean of the requests
 - **All_low, All_high:** assign the lowest/highest frequency requested
- Voltage set for entire domain based on highest frequency (max)



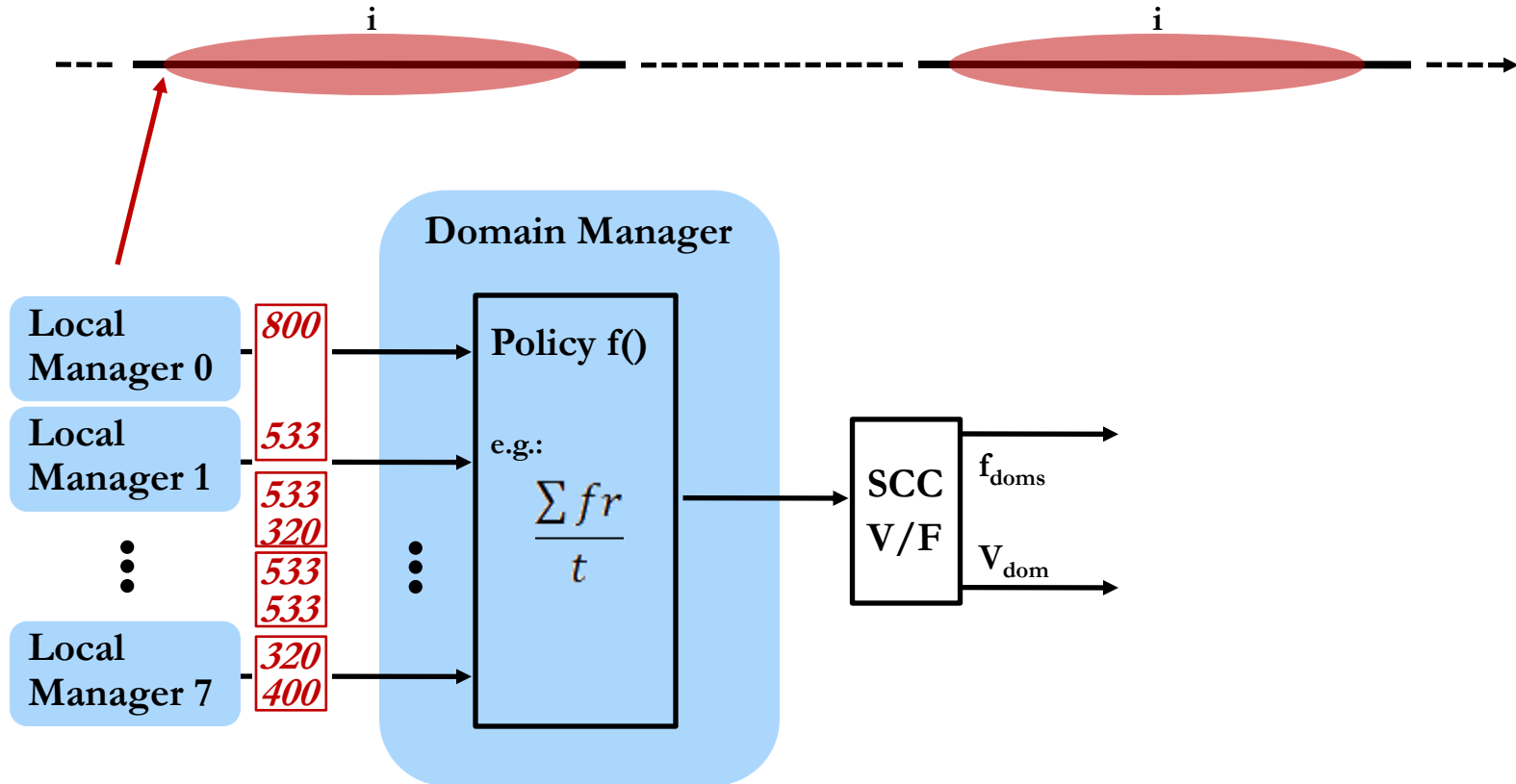
Domain Management Example



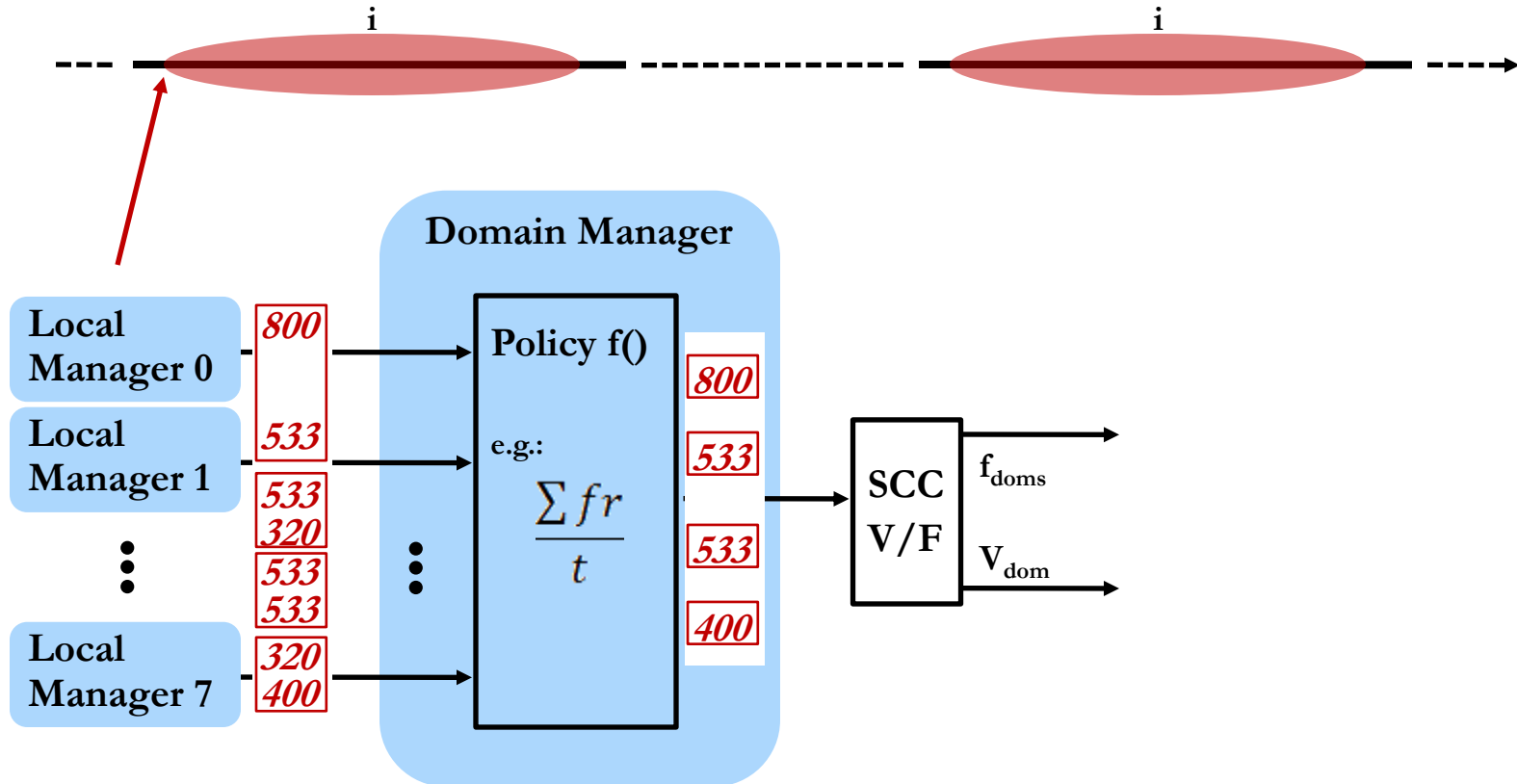
Domain Management Example



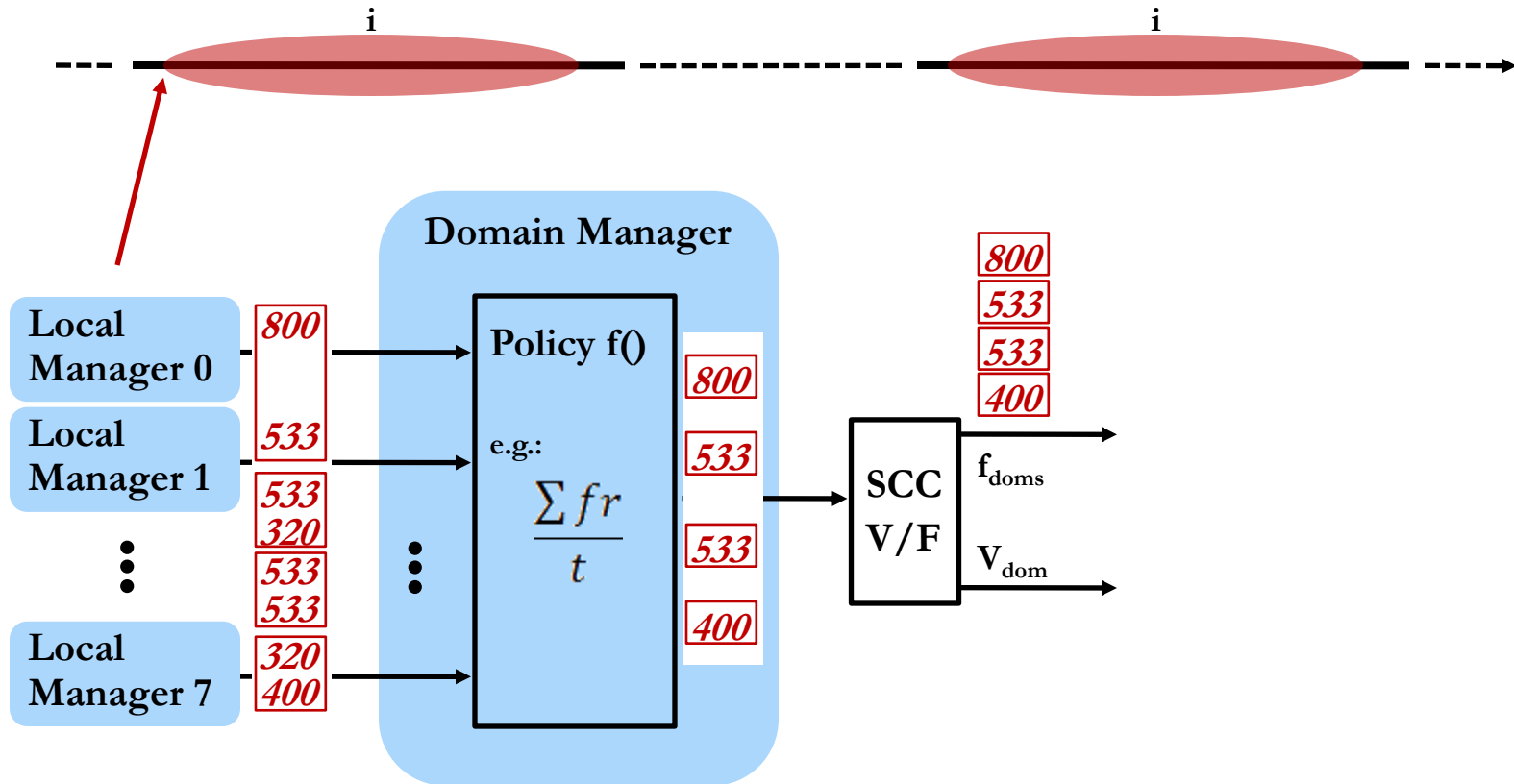
Domain Management Example



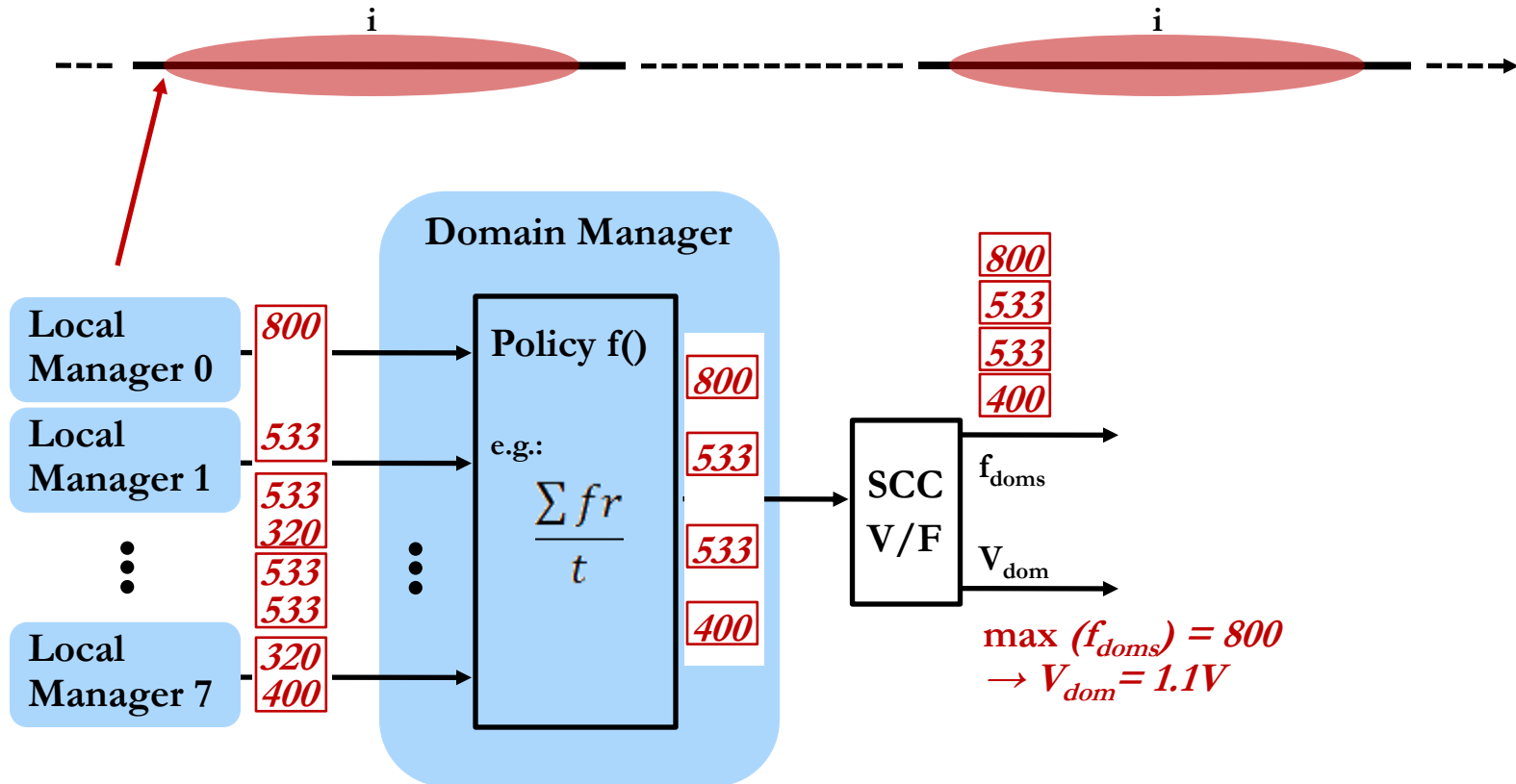
Domain Management Example



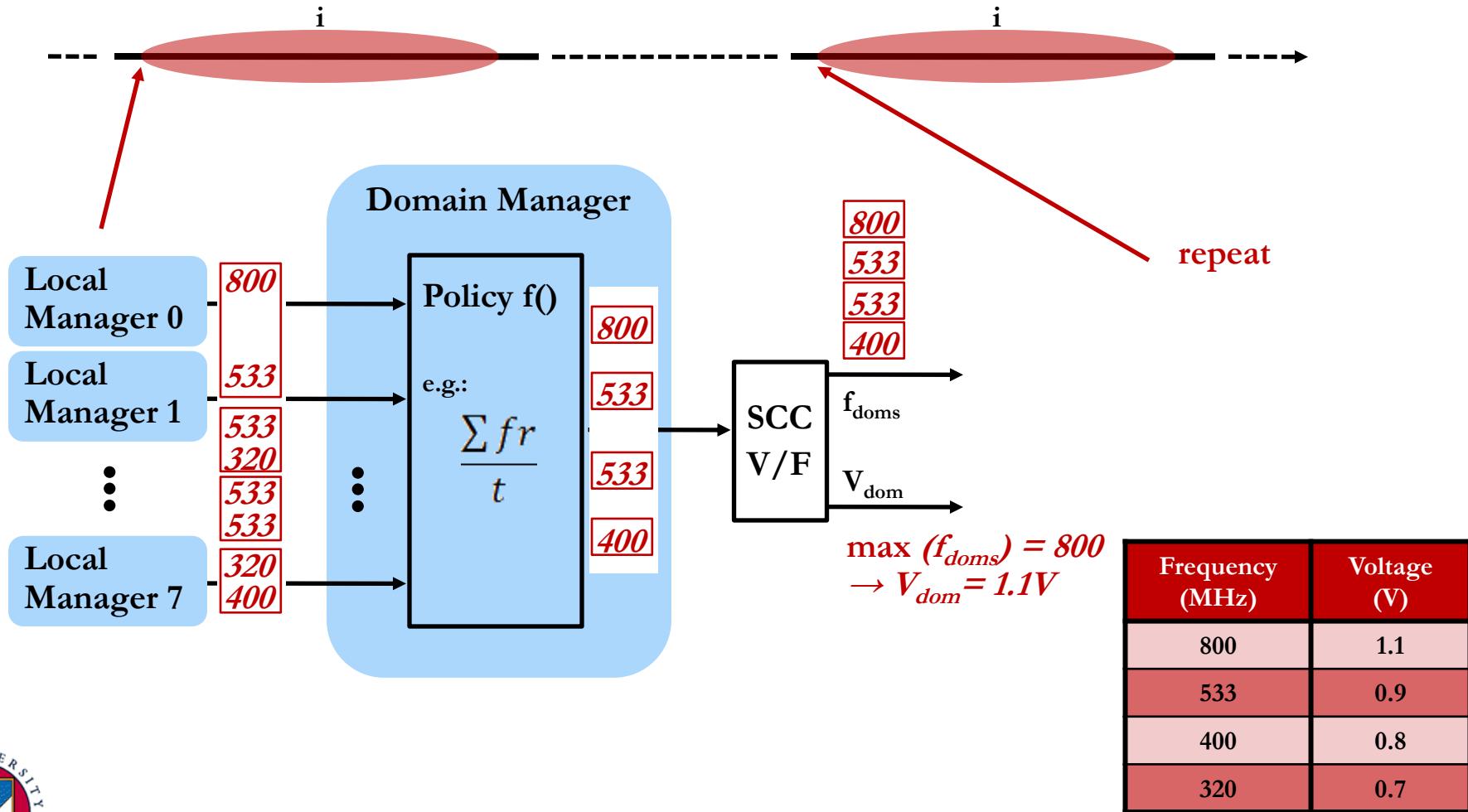
Domain Management Example



Domain Management Example



Domain Management Example



Outline

- Introduction
- Power Manager
- Phase Predictor
- Results
- Conclusions

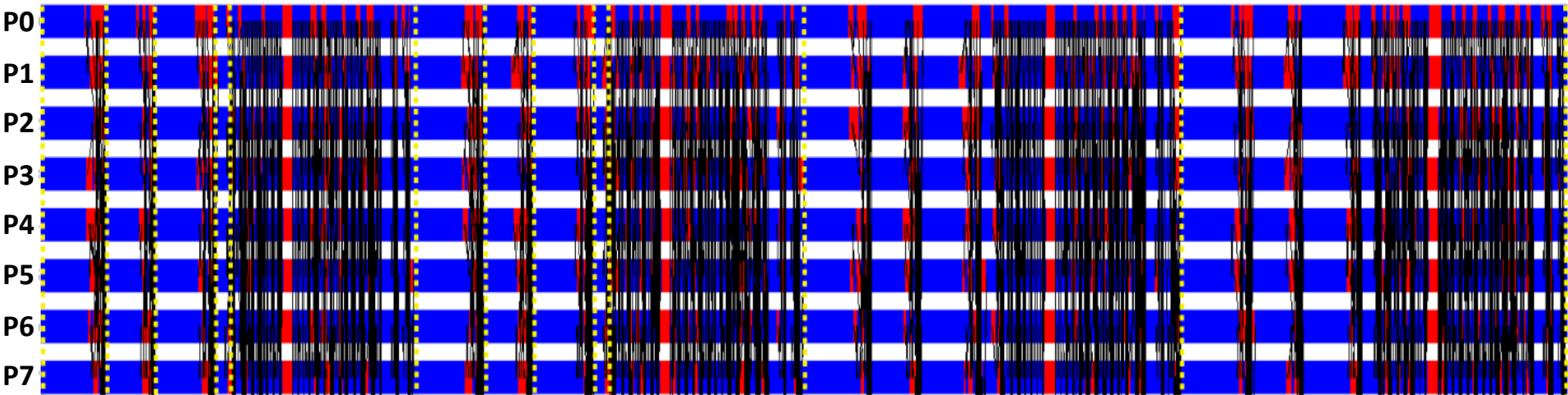


Phase Predictor

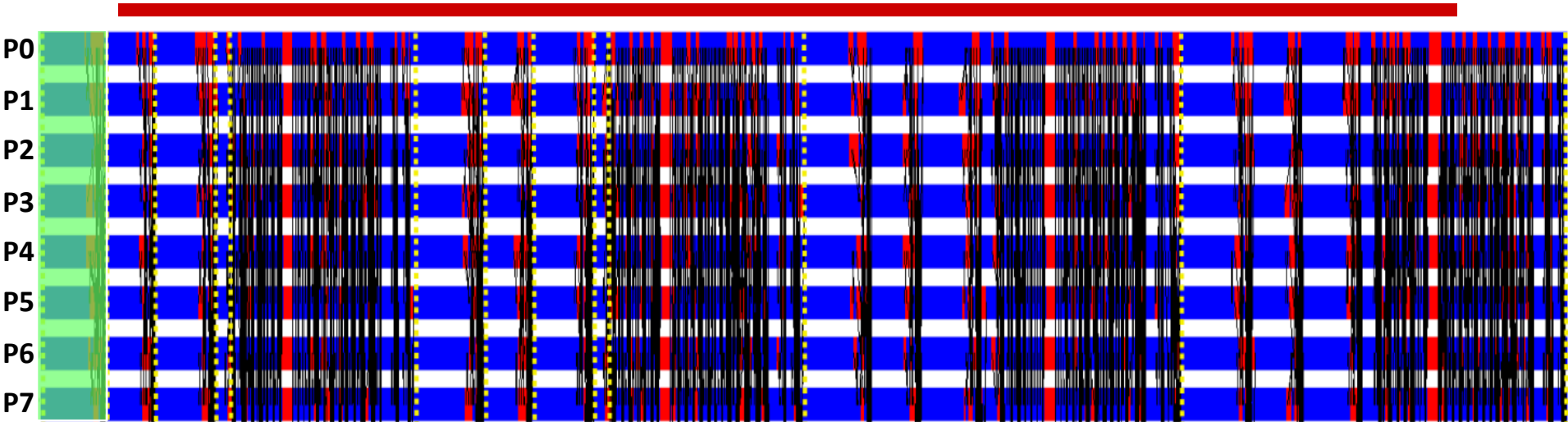
- Recurring patterns in MPI applications
 - Communication and execution patterns at MPI-event granularity
 - Patterns highly repeatable
- Pattern detection
 - Use of “Supermaximal Repeat String” algorithm
- Predictor
 - Predicts next call and program phase with projected execution time
 - Places DVFS scheduling points around repeatable regions within pattern
 - Front-end to the local power controllers
- Implemented as a wrapper library for MPI calls



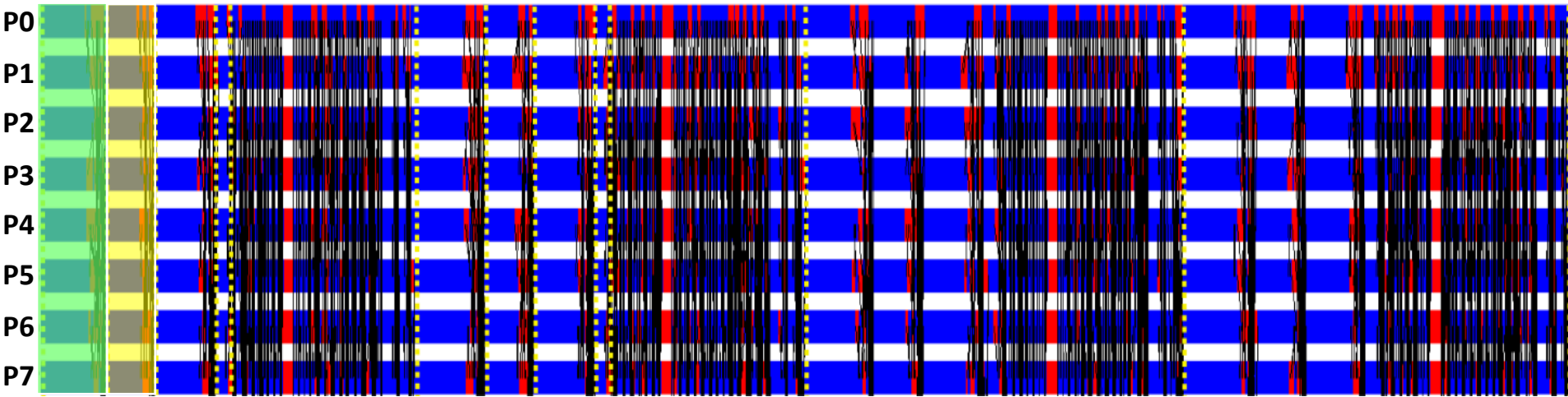
Phase Predictor Example



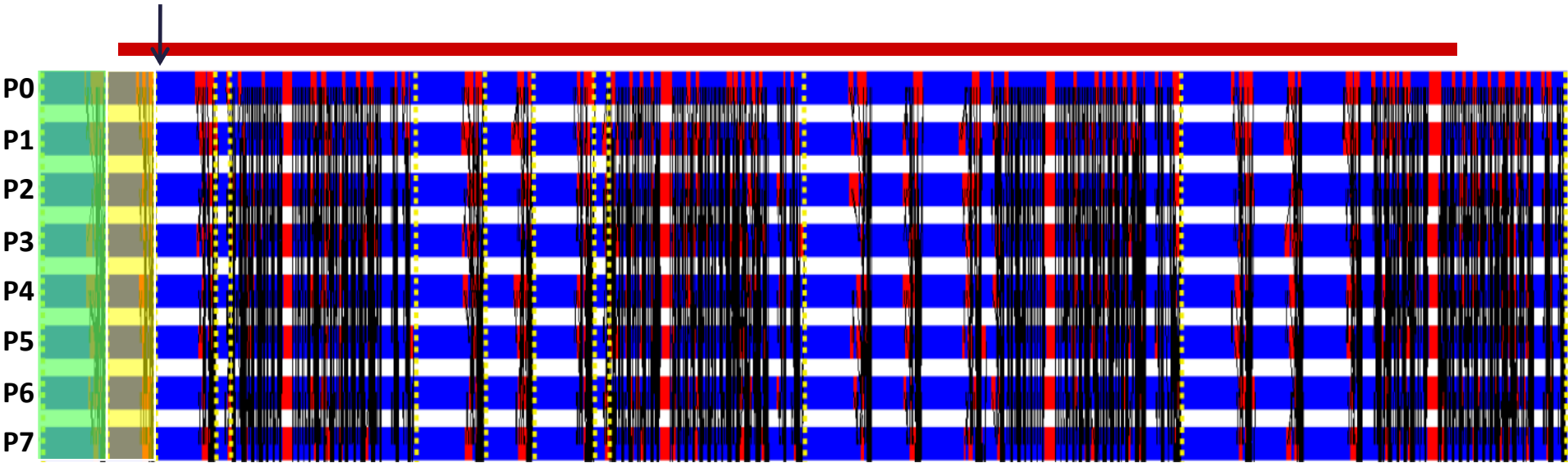
Phase Predictor Example



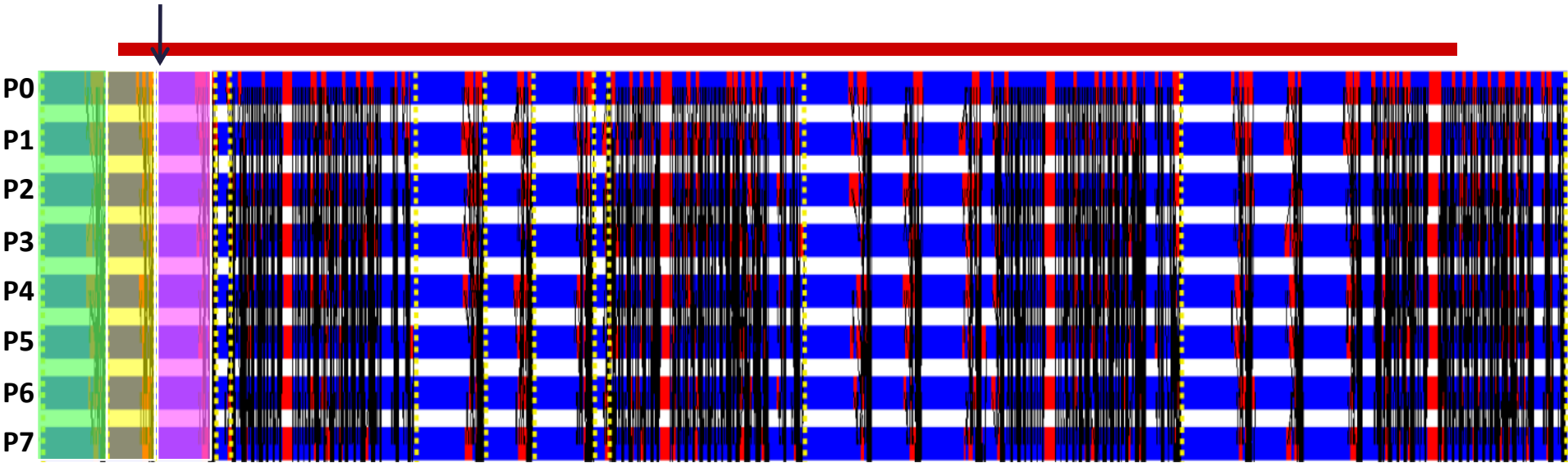
Phase Predictor Example



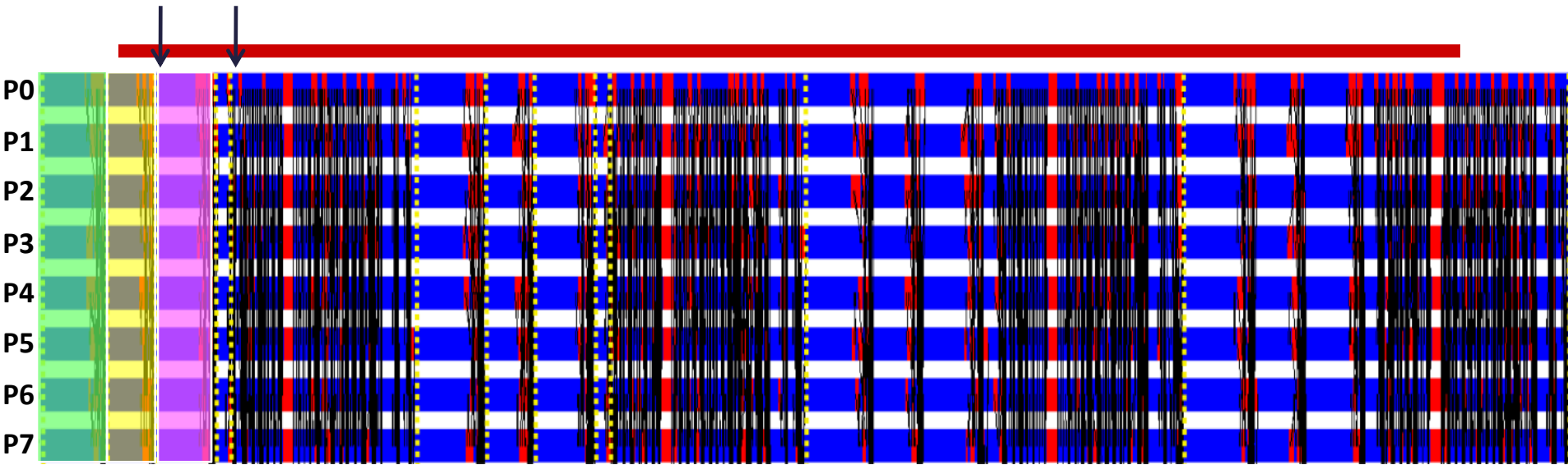
Phase Predictor Example



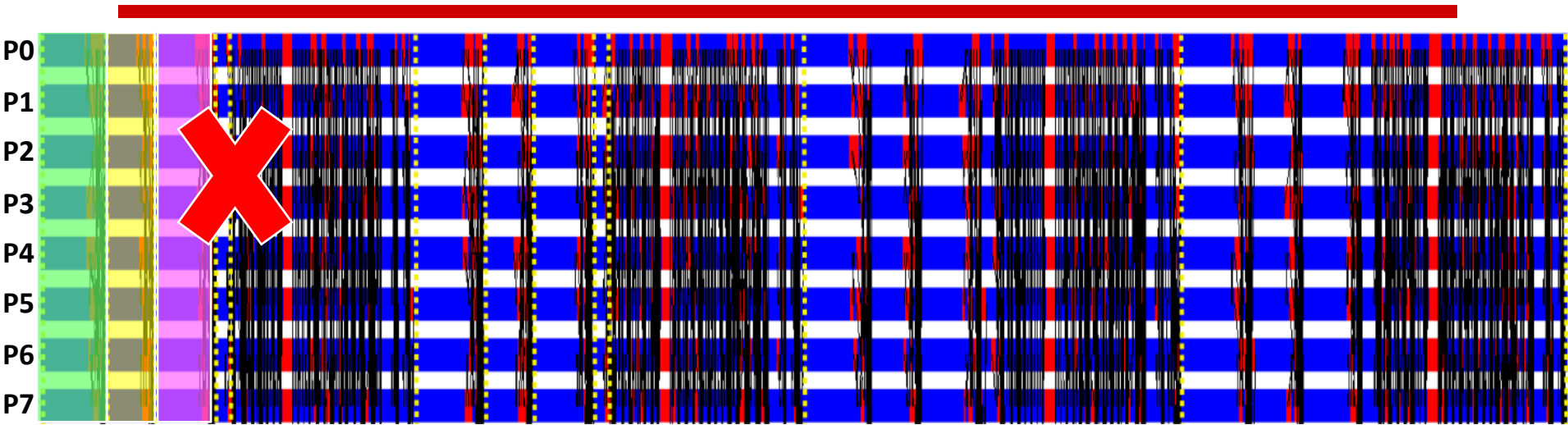
Phase Predictor Example



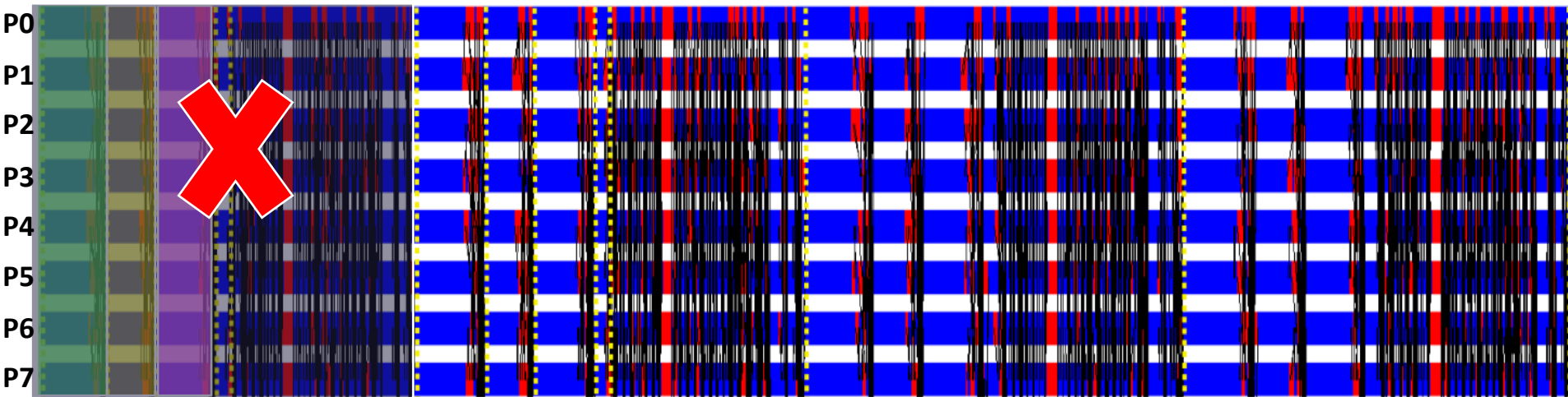
Phase Predictor Example



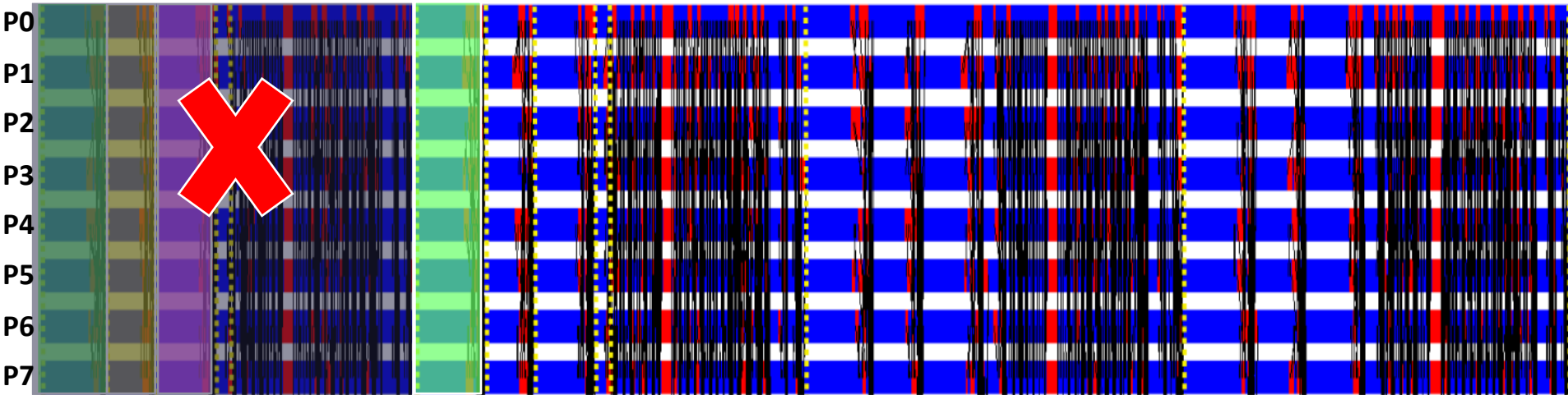
Phase Predictor Example



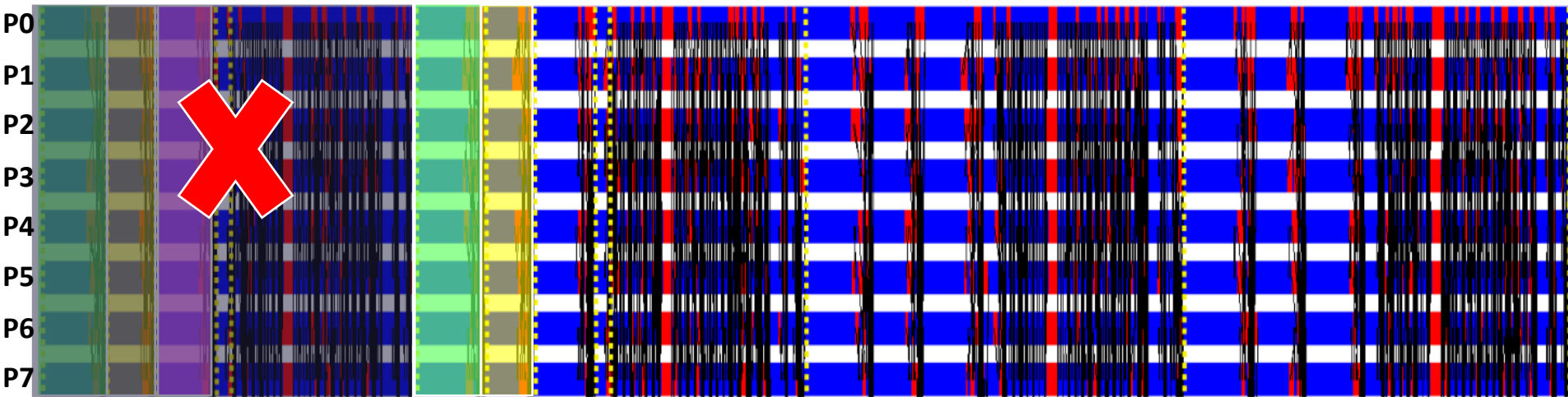
Phase Predictor Example



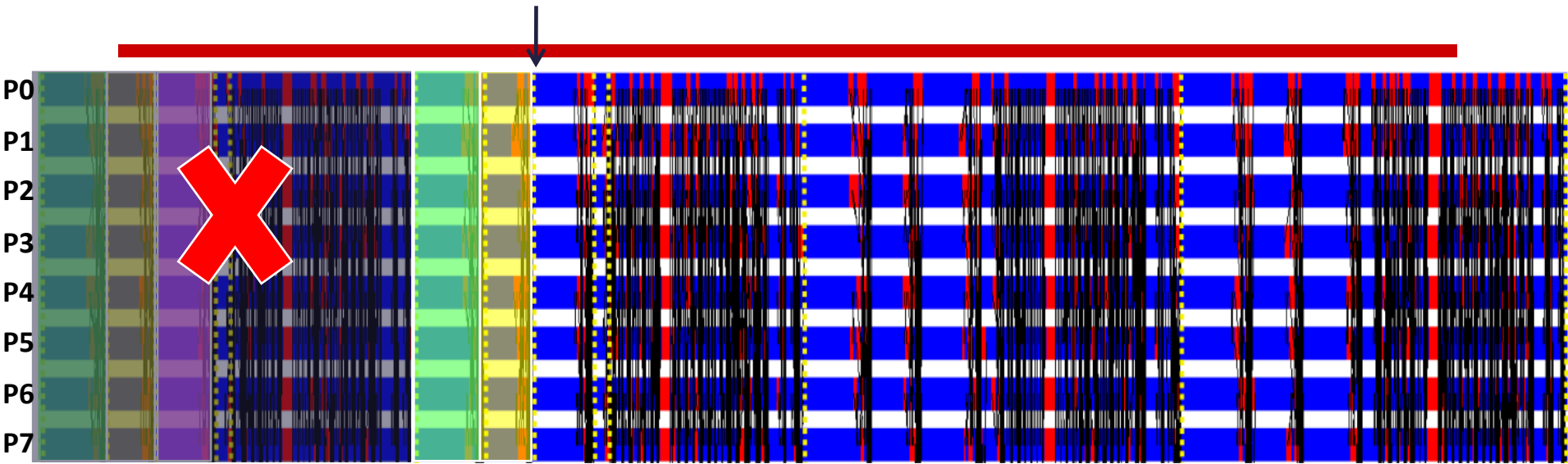
Phase Predictor Example



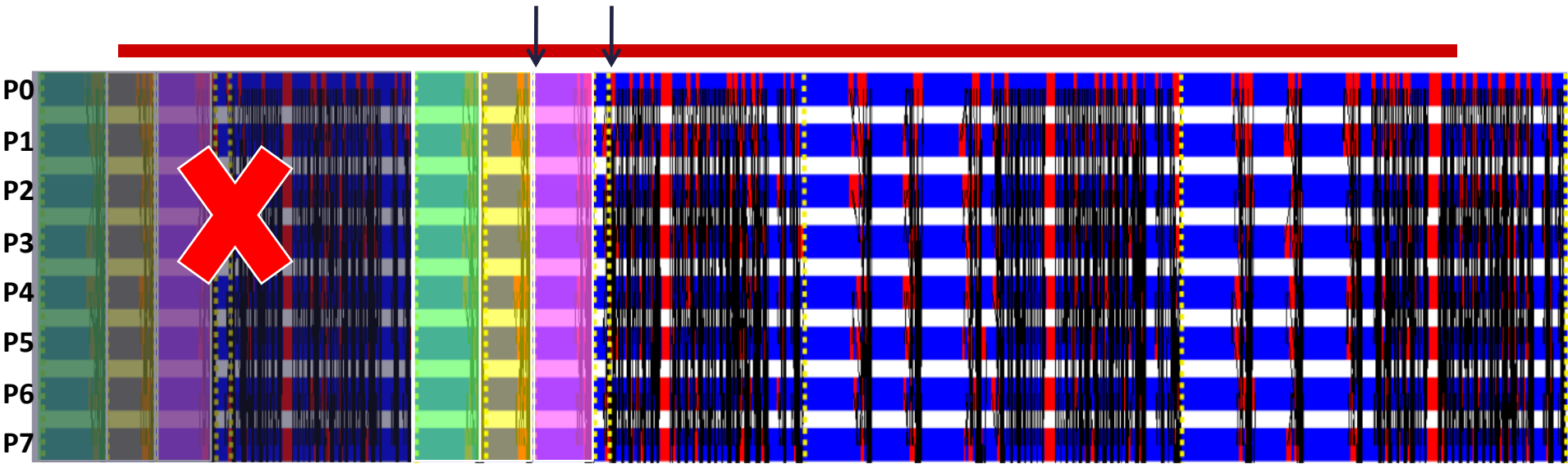
Phase Predictor Example



Phase Predictor Example



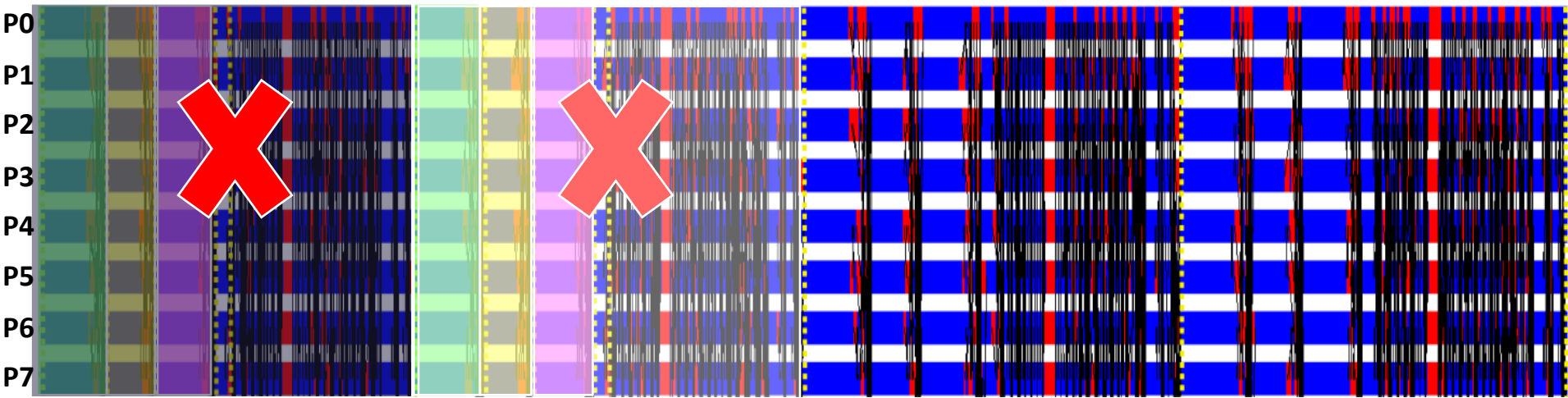
Phase Predictor Example



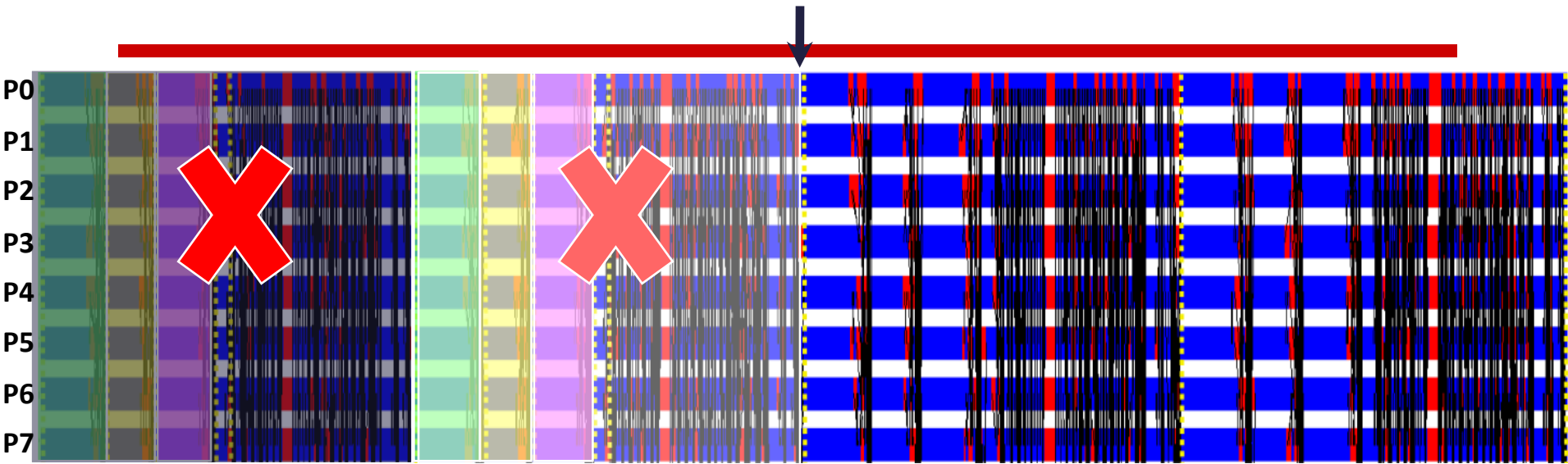
Phase Predictor Example



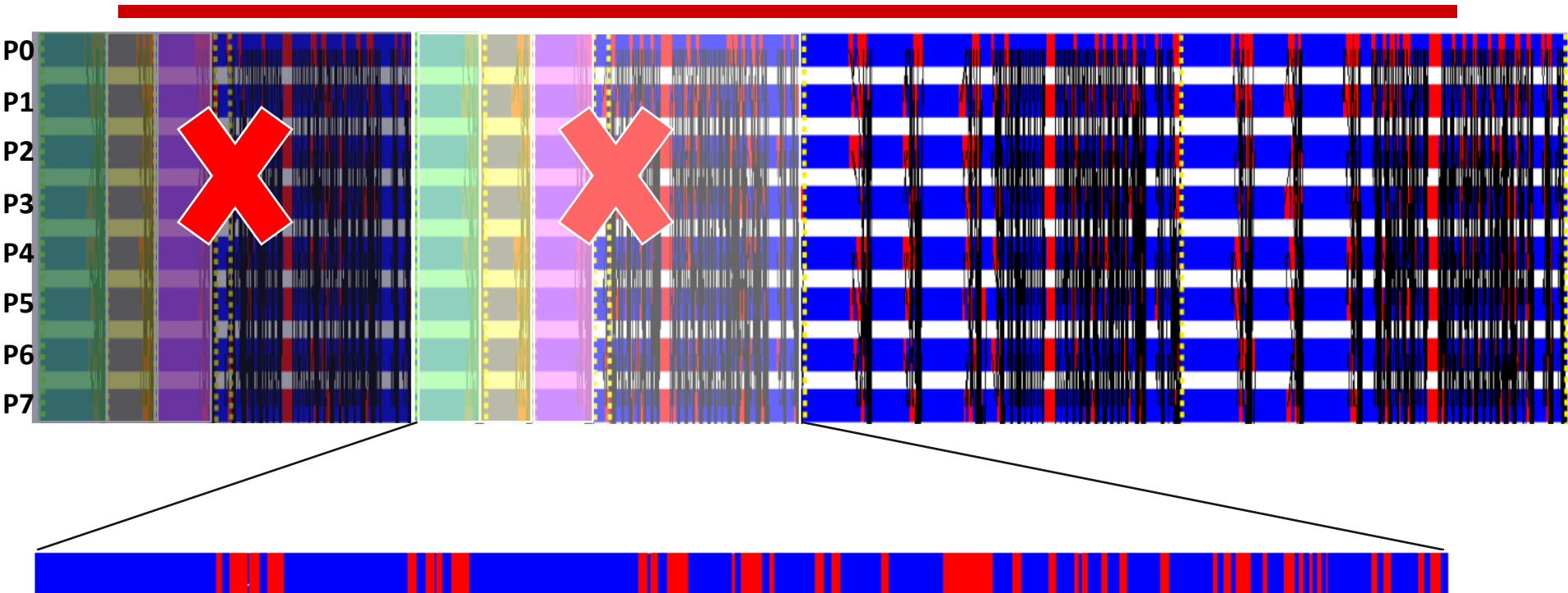
Phase Predictor Example



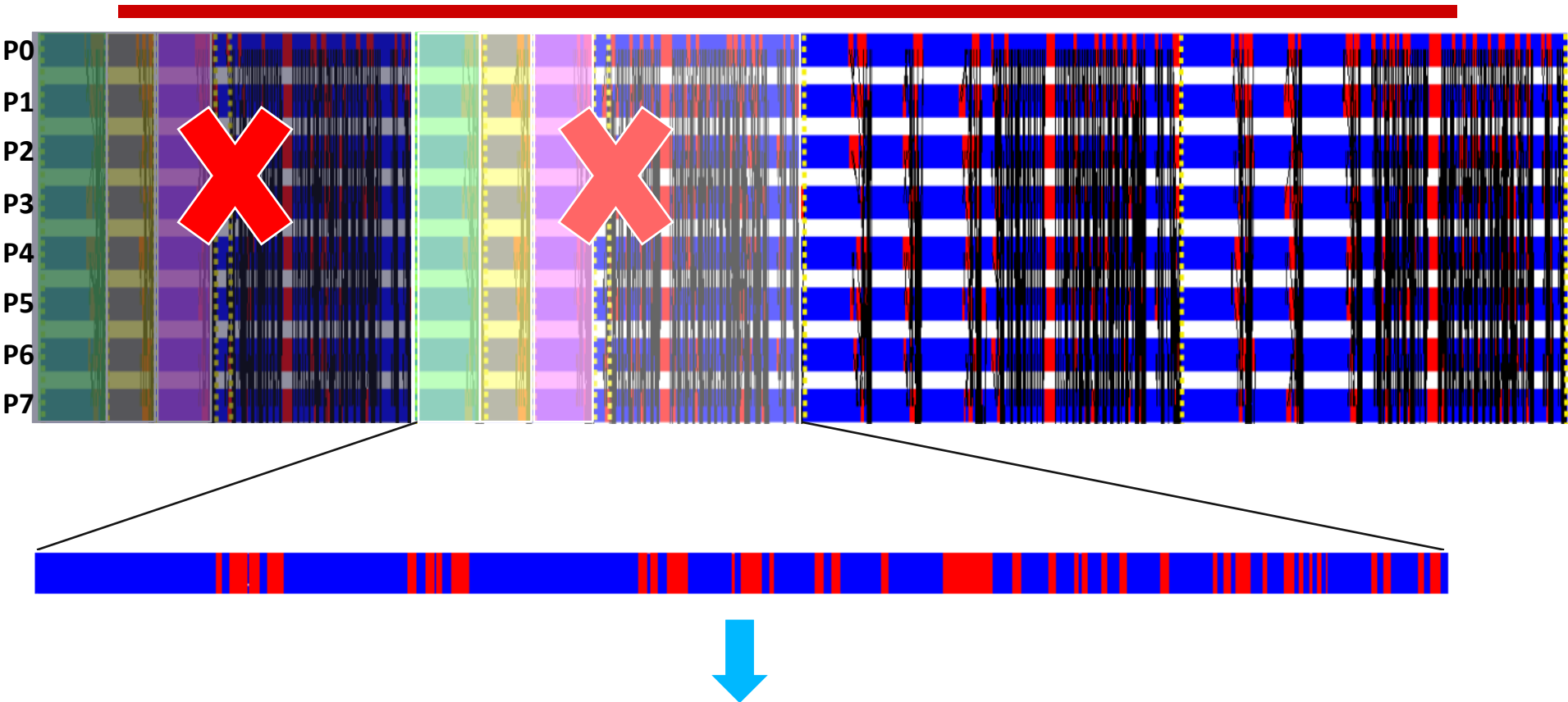
Phase Predictor Example



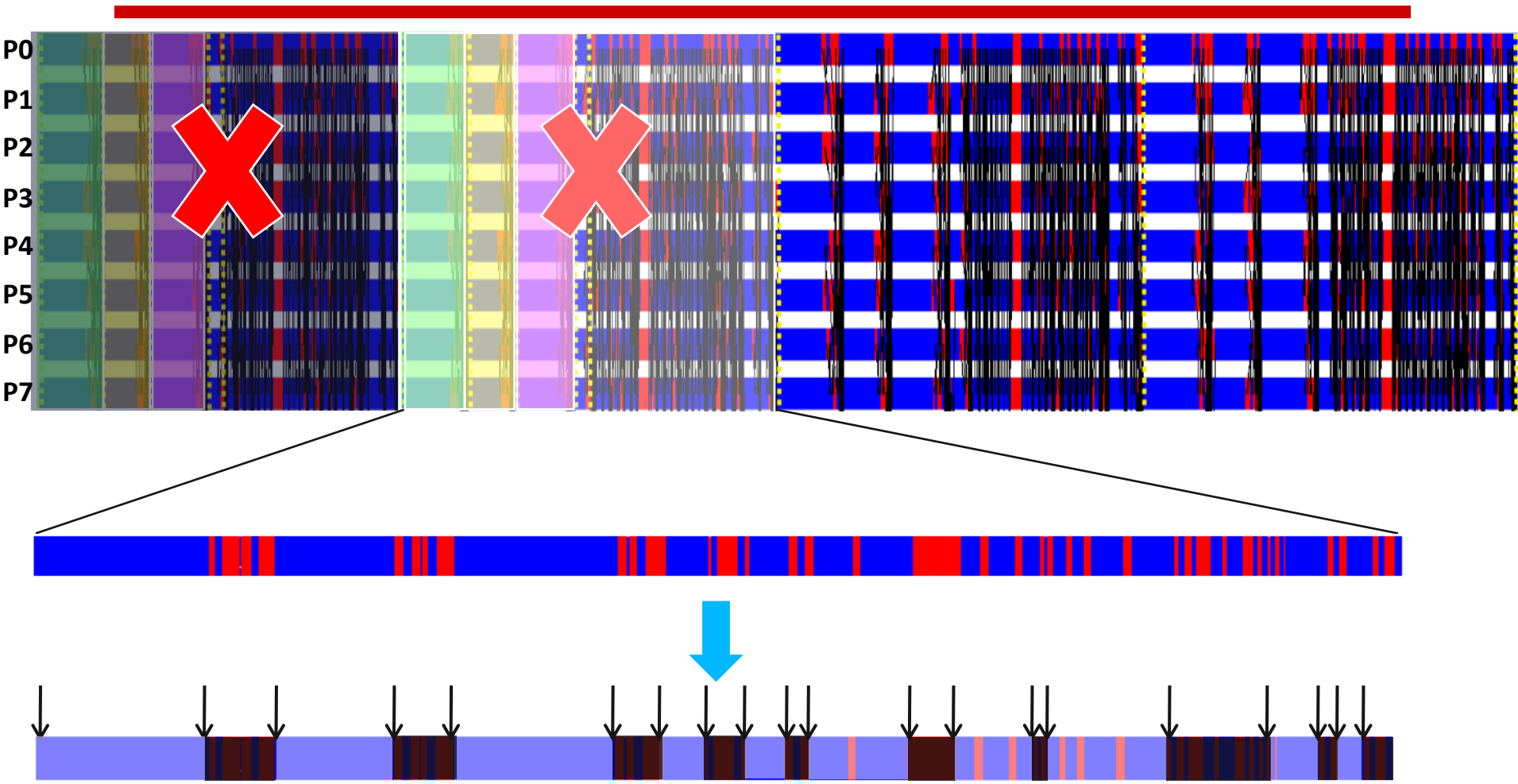
Phase Predictor Example



Phase Predictor Example



Phase Predictor Example



Outline

- Introduction
- Power Manager
- Phase Predictor
- Results
- Conclusions



Experimental Setup

- Platform

- Single chip Cloud Computer
- 48 cores running Linux kernel 2.6.16
- V and F levels obtained empirically

Frequency (MHz)	Voltage (V)
800	1.1
533	0.9
400	0.8
320	0.7

- Compiler:

- GCC 3.4.6

- Benchmarks:

- NAS MPI Parallel Benchmarks
- 2 SPEC MPI 2007

- Evaluation Methodology:

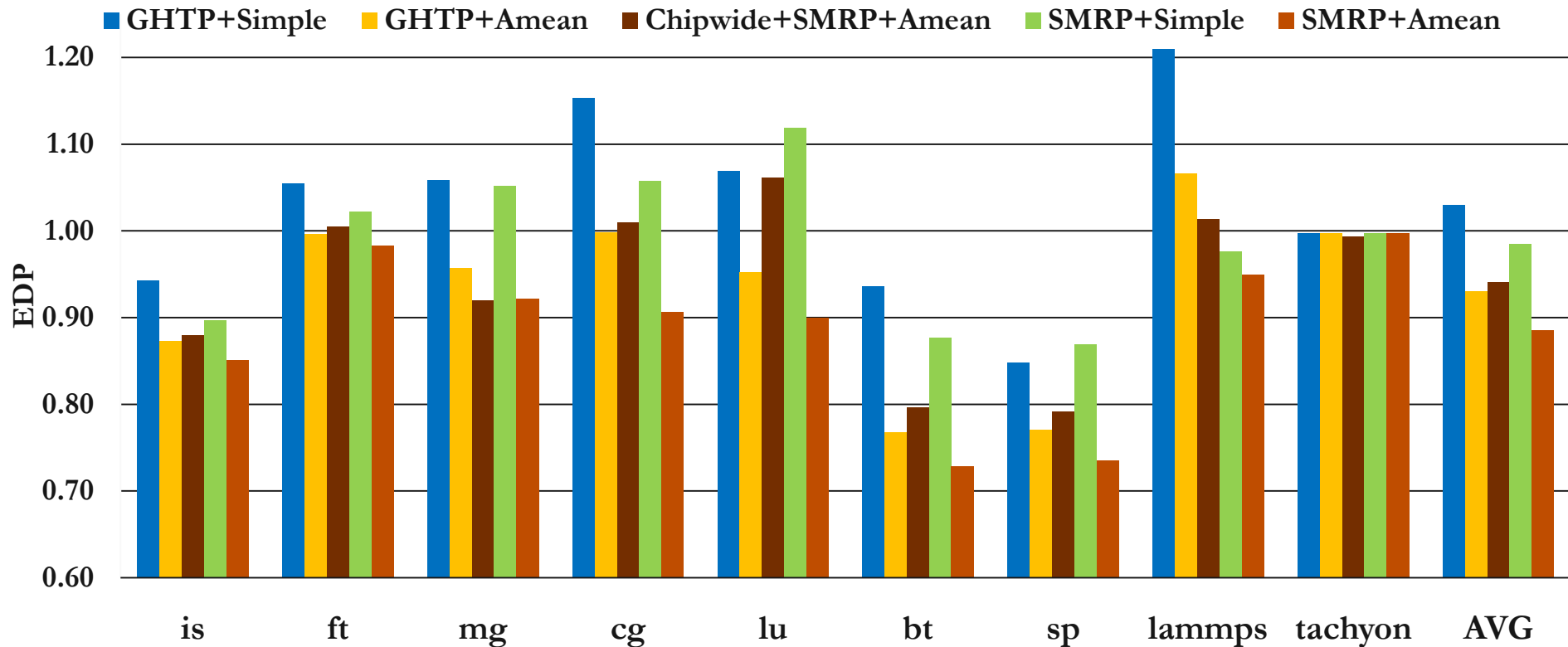
- Lab setup for accurate power measurement

- Schemes evaluated:

Scheme	Predictor	Domain Policy
SMRP + Amean	SMRP	Mean
SMRP + Simple	SMRP	Simple
Chipwide + SMRP + Amean	SMRP	Mean, but chip-wide
GHTP + Amean	GHTP	Mean
GHTP + Simple	GHTP	Simple



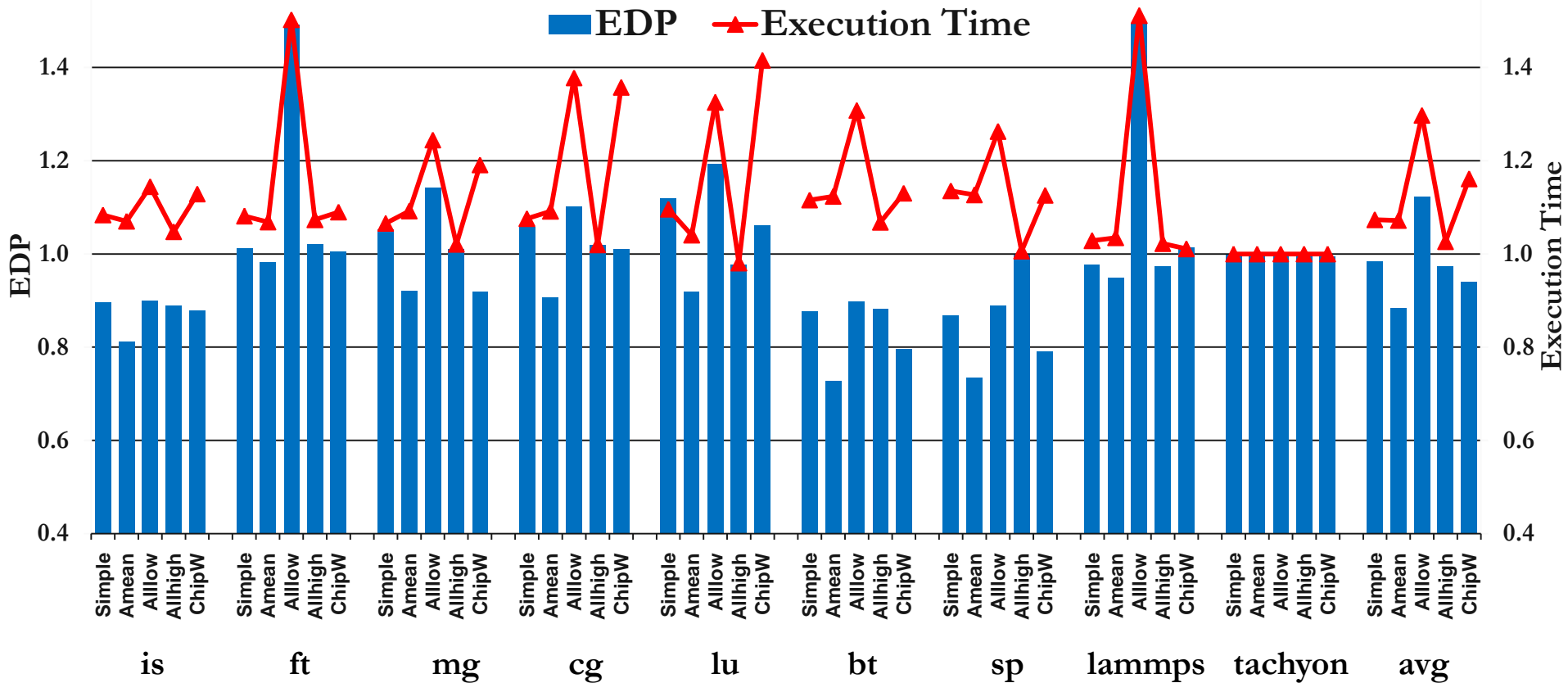
Results – Bottom Line



- 11% avg. EDP reduction with 7% increase in execution time



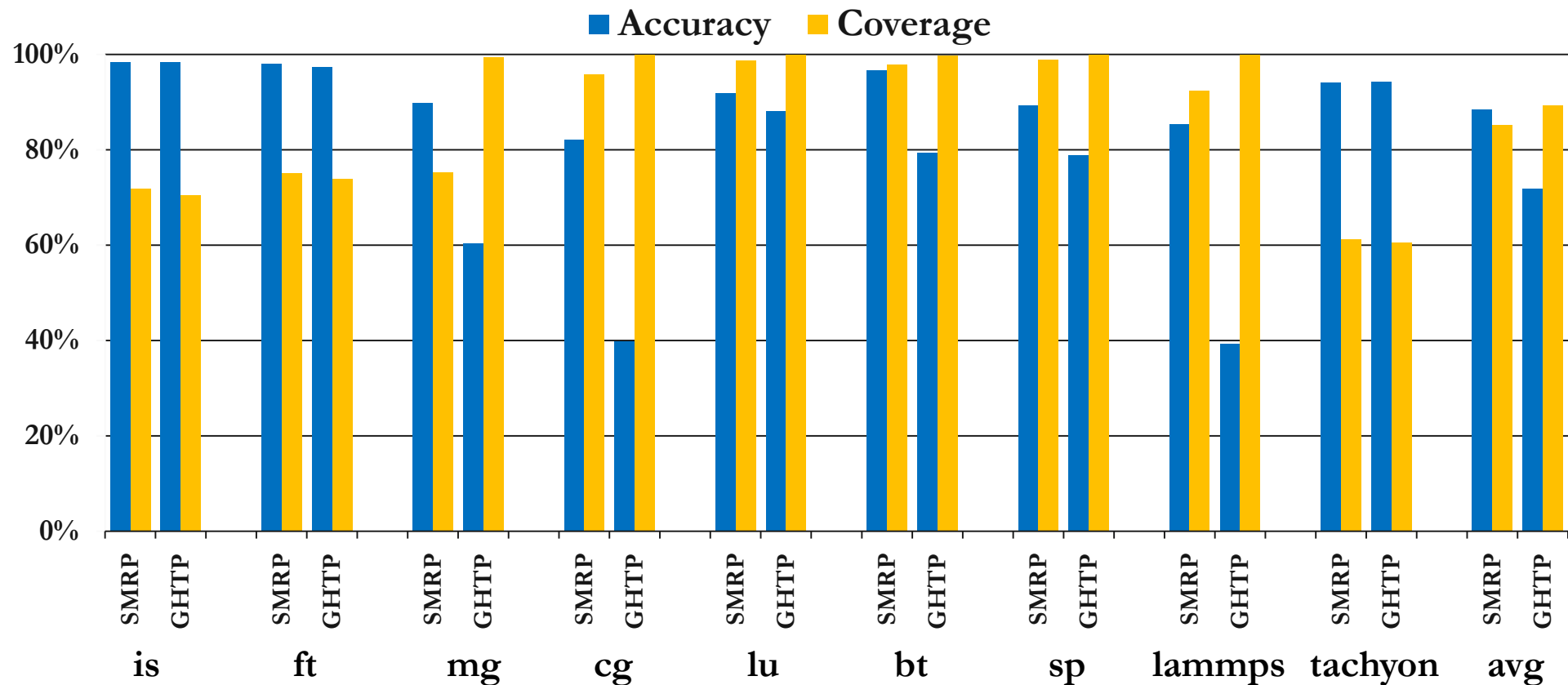
Results – Domain Management Policies



■ Arithmetic mean policy performs best



Results – Phase Predictor Performance



- Predictor 17% more accurate on avg. than state-of-the-art GHT



Conclusions

- Many-cores offer new challenges and opportunities in DVFS
 - Possible division into domains
 - Possible application-level control
- Presented a novel power mng. scheme applicable to many-cores
 - Modularity: allows for separation of concerns between phase detection/prediction and control
 - Hierarchical: can accommodate control at domains
 - Transparent: does not require user or OS intervention
- Demonstrated significant energy improvements of 15% on average on a real system
 - Benefits come from both better prediction and better management



Phase-Based Application-Driven Power Management on the Single-chip Cloud Computer

Nikolas Ioannou[†], Michael Kauschke[‡], Matthias Gries[‡], and
Marcelo Cintra[‡]

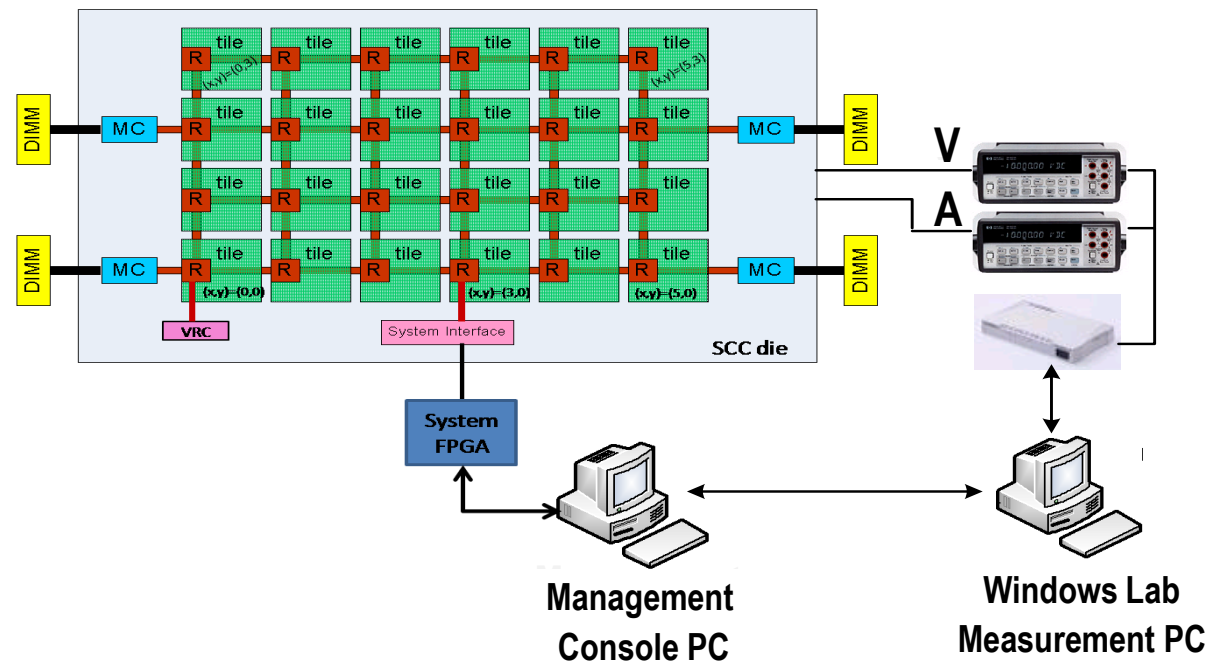
[†]University of Edinburgh

[‡]Intel Labs Braunschweig



Experimental Setup

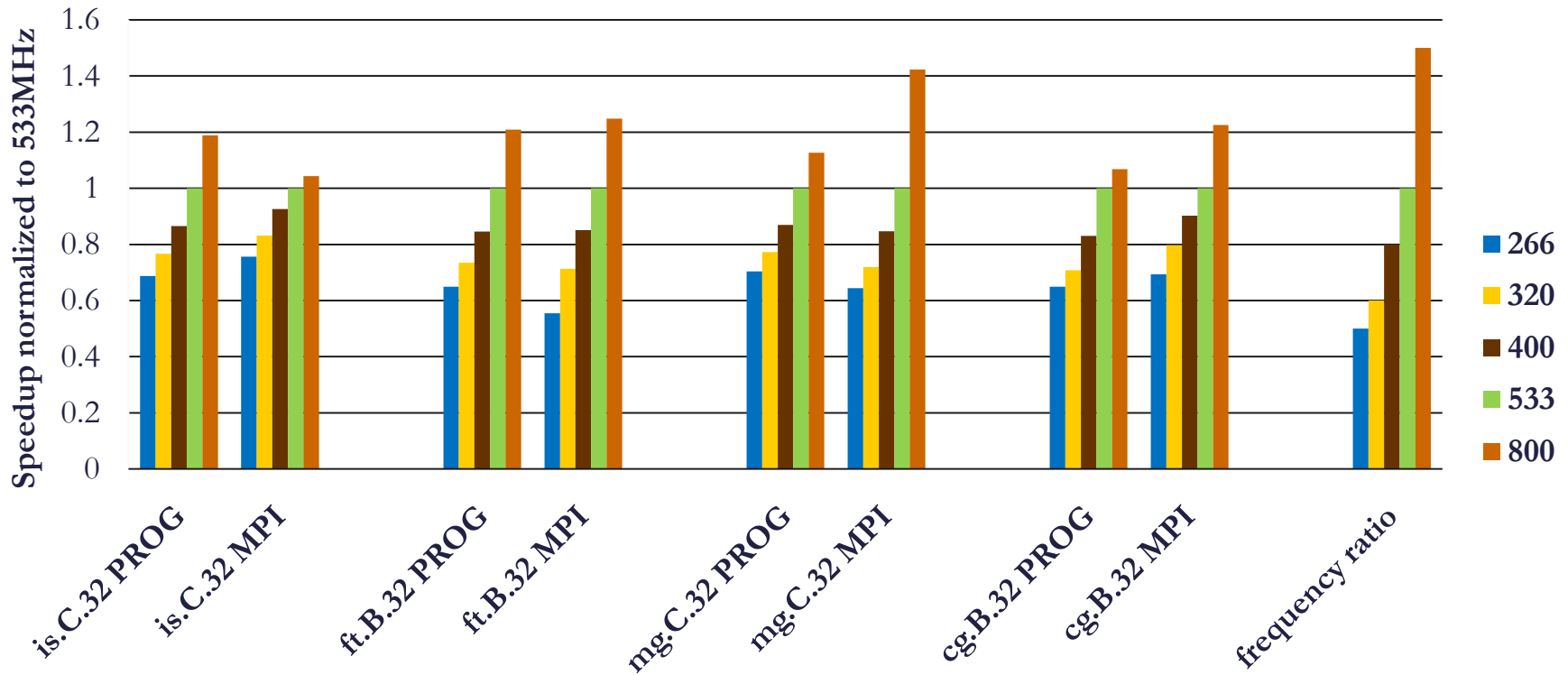
- Lab environment to accurately and directly measure system power
 - Directly measure input voltage and current with digital multimeters – current measure through shunt resistor



SCC MPI Frequency Scaling

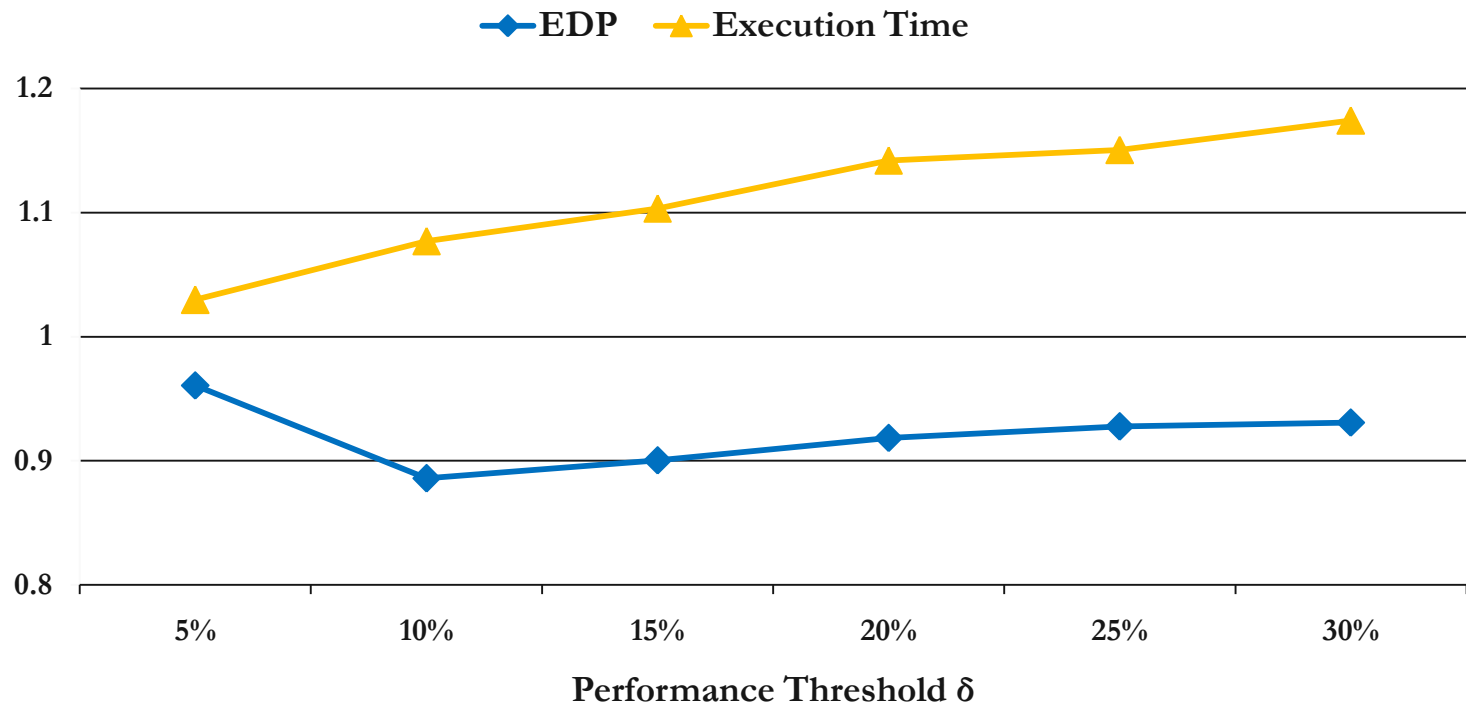
- SCC is less communication bound than traditional clusters

Computation and Communication Speedup Ratios for different Frequencies on the SCC



Performance Threshold Sensitivity

- 10% seems to be the sweet spot



Related Work

- Hardware schemes (Isci et. al. MICRO'06, Huang et. al. ISCA'03)
 - Require additional hardware for monitoring and control
 - Results obtained through simulation
- DVFS management of MPI applications (Freeh et. al. PPOPP'05, Lim et. al. SC'06, Rountree et. al. ICS'09)
 - Assume per-core power management
 - Require profile data
- Powernap (Meisner et. al. ASPLOS'09)
 - Idle time policy
 - Our scheme is complementary to such idle time schemes



Bibliography

- C. Isci, A. Buyuktosunoglu, C.-Y. Cher, and M. Martonosi
“An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget”
MICRO 2006
- M. C. Huang, J. Renau, and J. Torrellas
“Positional Adaptation of Processors: Application to Energy Reduction”
ISCA 2003
- V. W. Freeh and D. K. Lowenthal
“Using Multiple Energy Gears in MPI Programs on a Power-Scalable Cluster”
PPoPP 2005
- B. Rountree et. al.
“Adagio: Making DVS Practical for Complex HPC Applications”
ICS 2009



Bibliography (cont.)

- M. Y. Lim, V. W. Freeh, and D. K. Lowenthal
“Adaptive, Transparent Frequency and Voltage Scaling of Communication Phases in MPI Programs”
SC 2006
- D. Meisner, B. T. Gold, and T. F. Wenisch
“PowerNap: Eliminating Server Idle Power”
ASPLOS 2009



Background: DVFS

- Dynamic power consumption: $P_{dy} \propto V_{dd}^2 \cdot f$
- Frequency is also a function of V_{dd} (lower $V_{dd} \rightarrow$ lower f)
- Thus, lowering both V_{dd} and f can bring significant power savings
- If power savings come with little impact on performance then energy savings can be achieved as well
- DVFS usually applied to cores but not memories
 - Higher core $f \rightarrow$ higher memory latency in core cycles
- Commonly accepted rule-of-thumb is
 - 3:1 ratio of power savings to performance degradation

Background: MPI Applications

- Library supporting message-passing programming model
 - User API for exchanging messages across abstract processes
 - Common messages are send, receive and collective types
 - System interface to hardware communication mechanisms (e.g., TCP/IP, Infiniband, vendor proprietary)
- In most systems library runs in the same core as the user code
 - DVFS can be applied to both user code and MPI library code
- Common programming styles lead to much regularity in patterns of message exchanges
- Well-defined standard allows for easy addition of “wrappers” to common MPI calls



Background: SCC Many-Core

- Experimental concept vehicle developed by Intel Labs to serve as a platform for software research
- 48 Intel Pentium® IA cores
- Tiled organization with 2 cores per tile and mesh interconnect
- Frequency domain per tile and voltage domain per 4 tiles
- Current frequency and voltage level can be read and set by user software through registers
- Voltage changes take $\sim 1\text{ms}$ and frequency changes take only a few cycles

